2009-06

Propeller performance analysis using lifting line theory

Flood, Kevin M.

Cambridge Massachusetts Institute of Technology

Propeller Performance Analysis

Using Lifting Line Theory

by

Kevin M. Flood

B.S., (1997) Hobart College
M.A., (2002) Webster University

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer

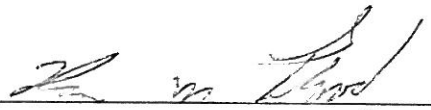
and

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Signature of
Author 

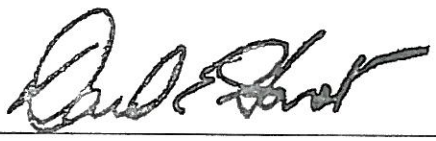
Department of Mechanical Engineering
May 8, 2009

Certified
by 

Mark S. Welsh
Professor of the Practice of Naval Construction and Engineering
Thesis Supervisor

Certified
by 

Richard W. Kimball
Thesis Supervisor

Accepted
by 

David E. Hardt
Chairman, Departmental Committee on Graduate Students
Department of Mechanical Engineering

Propeller Performance Analysis
Using Lifting Line Theory

by
Kevin M. Flood

Submitted to the Department of Mechanical Engineering on May 8, 2009
in Partial Fulfillment of the Requirements for the Degrees of
Naval Engineer
and
Master of Science in Mechanical Engineering

Abstract

Propellers are typically optimized to provide the maximum thrust for the minimum torque at a specific number of revolutions per minute (RPM) at a particular ship speed. This process allows ships to efficiently travel at their design speed. However, it is useful to know how the propeller performs during off-design conditions. This is especially true for naval warships whose missions require them to perform at a wide range of speeds. Currently the *Open-source Propeller Design and Analysis Program* can design and analyze a propeller only at a given operating condition (i.e. a given propeller RPM and thrust). If these values are varied, the program will design a new optimal propeller for the given inputs. The purpose of this thesis is to take a propeller that is designed for a given case and analyze how it will behave in off-design conditions.

Propeller performance is analyzed using non-dimensional curves that depict thrust, torque, and efficiency as functions of the propeller speed of advance. The first step in producing the open water diagram is to use lifting line theory to characterize the propeller blades. The bound circulation on the lifting line is a function of the blade geometry along with the blade velocity (both rotational and axial). Lerbs provided a method to evaluate the circulation for a given set of these conditions. This thesis implements Lerbs method using MATLAB® code to allow for fast and accurate modeling of circulation distributions and induced velocities for a wide range of operating conditions. These values are then used to calculate the forces and efficiency of the propeller. The program shows good agreement with experimental data.

Thesis Supervisor: Mark S. Welsh
Title: Professor of the Practice of Naval Construction and Engineering

Thesis Supervisor: Richard W. Kimball

Acknowledgements

The author thanks the following individuals for all of their support and assistance with this thesis:

Professor Rich Kimball for all his wisdom and guidance during not only the thesis process but also during the two classes he taught. His classes and teaching method were among the best experienced at MIT.

CAPT Patrick Keenan for the leadership and direction he provided throughout his time leading the course 2N program. His course was one of the most interesting the author experienced at MIT.

CAPT Mark Welsh for the leadership and support he provided in the "home stretch" of my MIT experience.

And most of all, my family for all of their support and understanding during my time at MIT. Missy, Margaret, Josh, and I had a wild ride filled with fun and frustration for the past three years.

Table of Contents

List of Figures

List of Tables

1. Introduction and Recent Propeller Design Advances

Propellers are typically optimized to provide the maximum thrust for the minimum torque at a specific number of revolutions per minute (RPM) at a particular ship speed. This process works very well for merchant ships that travel across the ocean at a relatively constant speed. There is a certain RPM at which their engines operate most efficiently. If their propeller can produce the thrust to achieve the desired speed while turning at the proper rate, then the ship will arrive at its destination using the minimal amount of fuel. The process is not much different for a Naval ship except that tactical situations often require a wider range of speeds to be used in order to accomplish the mission. Therefore, even though a ship's propeller can be optimized for a given set of conditions, it is very useful to know how the propeller performs at off-design conditions.

The purpose of this thesis is to take a propeller that is designed for a given case and analyze how it will behave at off-design advance ratio values. This will be done by producing the propeller open water diagram as described by Woud and Stapersma in (1). Propeller performance can be expressed in four non-dimensional parameters:

- The advance ratio, J_S.
- The thrust coefficient, K_T.
- The torque coefficient, K_Q.
- The open water efficiency, η.

The advance ratio non-dimensionalizes the ships velocity (V_S), which is the same as the velocity of advance (V_a) since there are no wake effects in an open water analysis, using the propeller tip speed as shown in equation (1.1). Thrust, T, and torque, Q, are also non-dimensionalized using the propeller's speed, n, and diameter, D, as well as the density of seawater, ρ, as shown in equations (1.2) and (1.3) respectively. The open water efficiency is expressed in terms of the above parameters as shown in equation (1.4). A propeller open water diagram is constructed such that K_T, K_Q, and η are all functions of J_S. In keeping with common practice, K_Q is multiplied by ten in order to better present the curves on one set of axes.

$$J_S = \frac{V_S}{nD} \tag{1.1}$$

$$K_T = \frac{T}{\rho n^2 D^4} \tag{1.2}$$

$$K_Q = \frac{Q}{\rho n^2 D^5} \tag{1.3}$$

$$\eta = \frac{1}{2\pi} \cdot \frac{K_T J_S}{K_Q} \tag{1.4}$$

The first step in producing the open water diagram is to use lifting line theory to characterize the propeller blades. In this theory the blade is replaced by a straight line. As described in Abbott (2), the circulation, Γ, about the blade associated with lift is replaced by a vortex filament. The vortex filament lies along the straight line; and, at each span-wise station, the strength of the vortex is proportional to the local intensity of lift. According to Helmholtz's theorem, a vortex filament cannot terminate in the fluid. The variation in vortex strength is therefore assumed to result from superposition of a number of horseshoe shaped vortices, as shown in Figure 1 from (3). The portions of the vortices lying along the span are called the bound vortices. The portions of the vortices extending downstream indefinitely are called the trailing or free vortices.

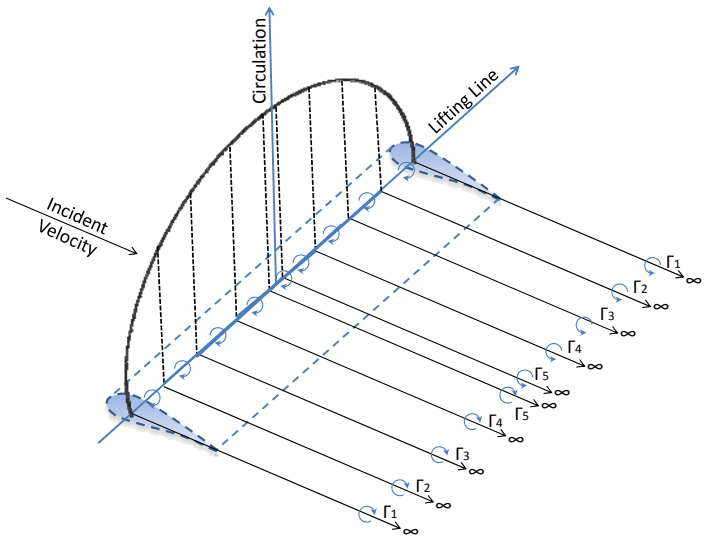


Figure 1: Vortex Pattern Representing a Lifting Wing (3)

The effect of the trailing vortices corresponding to a positive lift is to induce a downward component of velocity at and behind the blade. This downward component is called the downwash. The magnitude of the downwash at any section along the span is equal to the sum of the effects of all of the trailing vortices along the entire span. The effect of the downwash is to change the relative direction of the fluid stream over the section. The section is assumed to have the same hydrodynamic characteristics with respect to the rotated stream as it had in the normal two-dimensional flow. The rotation of the flow effectively changes the angle of attack. (2)

The method described by Lerbs in (4) is used to determine the circulation and induced velocity distributions for given values of J_S using the hydrodynamic properties of the foil sections at specified radial distances along the propeller blade. With the circulation and velocities calculated, functions built into the *Open-source Propeller Design and Analysis Program* are then used to determine the values of K_T, K_Q, and η. Finally, the whole process is repeated again for another value of J_S until the entire diagram is produced.

1.1 Recent Advances

Open-source Propeller Design and Analysis Program (OpenProp) is an open source MATLAB®-based suite of propeller numerical design tools. This program is an enhanced version of the *MIT Propeller Vortex Lattice Lifting Line Program* (PVL) developed by Professor Justin Kerwin at MIT in 2001. OpenProp v1.0, originally titled MPVL, was written in 2007 by Hsin-Lung Chung and Kate D'Epagnier and is described in detail in (3) and (5). Two of its main improvements versus PVL are its intuitive graphical user interfaces (GUIs) and greatly improved data visualization which includes graphic output and three-dimensional renderings. OpenProp v2.0 was written in 2008 by John Stubblefield and is described in detail in (6). Version 2 adds the ability to model an axially symmetric ducted-propeller with no gap between the duct and the propeller. OpenProp v3.0 was written by Brenden Epps in 2009 and brings the capability to design a turbine as well as a more generalized optimization routine.

OpenProp was designed to perform two primary tasks: parametric analysis and single propeller design. Both tasks begin with a desired operating condition defined primarily by the required thrust, ship speed, and inflow profile. The parametric analysis produces efficiency diagrams for all possible combinations of number of blades, propeller speed, and propeller diameter for ranges and increments entered by the user. The efficiency diagrams are then used to determine the optimum propeller parameters for the desired operating conditions given any constraints (e.g. propeller speed or diameter) specified by the user. The single propeller design routine produces a complete propeller design which is optimized for the desired operating condition and defined propeller parameters (number of blades, propeller speed, propeller diameter, hub diameter, etc). If these values are varied, the program will design a new optimal propeller for those given inputs, but it will not analyze the same propeller at new conditions.

In 2008 Christopher Peterson developed MATLAB® executables that interface with a modified version of XFOIL for determining the minimum pressure of a foil operating in an inviscid fluid which are described in detail in (7). XFOIL is an analysis and design system for Low Reynolds Number Airfoils. XFOIL uses an inviscid linear-vorticity panel method with a Karman-Tsien compressibility correction for direct and mixed-inverse modes. Source distributions are superimposed on the airfoil and wake permitting modeling of viscous layer influence on the potential flow. Both laminar and turbulent layers are treated with an e^9-type amplification formulation determining the transition point. The boundary layer and transition equations are solved simultaneously with the inviscid flow field by a global Newton method as described by Drela in (8). Peterson's code creates minimum pressure envelopes, similar to those published by Brockett (1965). The modified XFOIL and MATLAB® interface was intended for future interface with OpenProp.

2. Theoretical Foundation

Each individual blade of a propeller can be represented by its own lifting line as described in Section 1. The downwash affects not only the blade being modeled, but also every other blade on the propeller. However, if the propeller is symmetrical, these affects will cancel each other out. Therefore, the free votices that are shed from the lifting line will be identical and spaced at constant angles to each other. The free vortex lines shed from the lifting line are not acted on by forces. Their directions, according to wing theory, coincide with that of the resultant lifting system. This results in the free vortex lines of a propeller forming a general helical shape. When all of the free vortex lines are combined, they form a free vortex sheet that is in the shape of a general helical surface. The shape of the helical sheets and their induced velocities are mutually dependant so certain assumptions about the shape of the helical sheets are required. The methods described in this thesis are applicable to moderately loaded propellers, therefore the following assumptions apply. The induced velocities are allowed to influence the shape of the free helical sheets. However, the effects of centrifugal forces and wake contraction on the shape of the sheets are neglected. The work of Betz, Lock, and Kramer showed that for a moderately loaded propeller with an optimum circulation distribution, the assumption that the deformation of the helical sheets in the axial direction can be neglected produced results in close agreement with experimental data. The assumption that the axial deformation can be neglected will be extended here to non-optimum circulation distributions. Therefore, the vortex sheets discussed in the upcoming sections are of general helical shape and are made up of cylindrical vortex lines of a constant diameter and pitch angle in the axial direction. In addition, the curling up of the vortex sheets a certain distance behind the propeller caused by the sheet's self motion is also disregarded. (4)

2.1 Velocity Field of Symmetrically Spaced Helical Vortex Lines

There are two possible ways to determine the velocity field of symmetrically spaced helical vortex lines, the integral by Biot-Savart and Laplace's differential equation. Kawada in (9) used the Laplace equation approach to develop analytical expressions for the velocity potential. These

expressions provide numerical results more readily than the elaborate numerical integration carried out by Strscheletzky in (10) following the Biot-Savart method. (4)

Kawada considers a propeller of g blades modeled as a symmetric system of g helical tip vortices combined with an axial hub vortex which has the same strength as all of the tip vortices combined. Kawada's potential function is only justified when assuming that the vortex system is infinitely long in both directions of the axis. By subtracting the potential of the hub vortex from this expression, the potential for an infinitely long symmetrical system of g helical vortices of constant radius, r_o, and constant pitch angle, β_{io}, is obtained. First, a cylindrical coordinate system (z, ψ, r) is defined where z equals zero at the propeller and is positive in the direction of flow, and ψ equals zero at the first lifting line. Then the potential for internal points ($r< r_o$) in the field is defined by equation (2.1), and the potential for external points ($r>r_o$) is defined by equation (2.2). (4)

$$\phi_i = \frac{g}{2}\overline{\Gamma} + \frac{g}{2\pi}\overline{\Gamma}\left\{\frac{z}{k_o} + 2\frac{r_o}{k_o}\sum_{n=1} I_{ng}\left(\frac{ng}{k_o}r\right)K'_{ng}\left(\frac{ng}{k_o}r_o\right)\sin ng\vartheta\right\} \qquad (2.1)$$

$$\phi_e = \frac{\overline{\Gamma}}{2\pi}\left\{g\psi - 2\pi\sum_{m=1}^{g}\frac{m-1}{g} + 2g\frac{r_o}{k_o}\sum_{n=1} K_{ng}\left(\frac{ng}{k_o}r\right)I'_{ng}\left(\frac{ng}{k_o}r_o\right)\sin ng\vartheta\right\} \qquad (2.2)$$

For the potential equations, $\overline{\Gamma}$ is defined as the infinitesimal circulation of one of the g vortex lines which is related to the circulation at one of the lifting lines, Γ, by equation (2.3). In addition, k_o and ϑ are defined in equations (2.4) and (2.5), respectively. The functions I and K are modified Bessel functions of the first and second kind. The prime symbol denotes the derivative with respect to the argument. (4)

$$\overline{\Gamma} = \frac{d\Gamma}{dr}dr \qquad (2.3)$$

$$k_o = r_o \tan(\beta_{io}) \qquad (2.4)$$

$$\vartheta = \psi - \frac{z}{k_o} \qquad (2.5)$$

In order to find the induced velocities at a lifting line of the propeller, equations (2.1) and (2.2) can be used with ψ and z both equal to zero. However, these velocities would correspond to a vortex system that extends from z equals positive infinity to negative infinity which is essentially a two-dimensional system. The vortex system of an actual propeller is three-dimensional and the vortex system extends only from z equals zero to positive infinity. In order to determine if velocities of the three-dimensional system could be determined from the two-dimensional system, Lerbs (4) considered the velocity-integrals by Biot-Savart for the assumed case that both r_o and β_{io} are independent of ψ. It was concluded from these integrals that the effect of the axial and tangential components of the system between z equals negative infinity and the propeller (z=0) is equal to the effect of the system between the propeller and z equals positive infinity for points on the lifting line. Therefore it follows that the axial, $\overline{w}_a$, and tangential, $\overline{w}_t$, velocities induced at a lifting line of a propeller are simply half the axial and tangential velocities induced from the potentials in equations (2.1) and (2.2). These induced velocities are shown for both the internal and external points in equations (2.6) through (2.9).

$$\overline{w}_{ai} = \frac{1}{2}\frac{\partial \phi_i}{\partial z} = \frac{g\overline{\Gamma}}{4\pi}\frac{1}{k_o}\left\{1 - 2g\frac{r_o}{k_o}\sum_{n=1} nI_{ng}\left(\frac{ng}{k_o}r\right)K'_{ng}\left(\frac{ng}{k_o}r_o\right)\right\} \tag{2.6}$$

$$\overline{w}_{ae} = \frac{1}{2}\frac{\partial \phi_e}{\partial z} = -\frac{g^2\overline{\Gamma}}{2\pi}\frac{r_o}{k_o^2}\sum_{n=1} nK_{ng}\left(\frac{ng}{k_o}r\right)I'_{ng}\left(\frac{ng}{k_o}r_o\right) \tag{2.7}$$

$$\overline{w}_{ti} = \frac{1}{2}\frac{1}{r}\frac{\partial \phi_i}{\partial \psi} = \frac{1}{r}\frac{g^2\overline{\Gamma}}{2\pi}\frac{r_o}{k_o}\sum_{n=1} nI_{ng}\left(\frac{ng}{k_o}r\right)K'_{ng}\left(\frac{ng}{k_o}r_o\right) \tag{2.8}$$

$$\overline{w}_{te} = \frac{1}{2}\frac{1}{r}\frac{\partial \phi_e}{\partial \psi} = \frac{1}{r}\frac{g\overline{\Gamma}}{4\pi}\frac{1}{k_o}\left\{1 + 2g\frac{r_o}{k_o}\sum_{n=1} nK_{ng}\left(\frac{ng}{k_o}r\right)I'_{ng}\left(\frac{ng}{k_o}r_o\right)\right\} \tag{2.9}$$

For the purposes of numerical integration, the Bessel functions and their derivatives are estimated by Nicholson's asymptotic expansions (11). The final results for the induced velocities at one of the lifting lines are shown in equations (2.10) through (2.13).

$$\overline{w}_{ai} = \frac{1}{k_o}\frac{g\overline{\Gamma}}{4\pi}\left(1 + B_2\right) \tag{2.10}$$

$$\overline{w}_{ae} = -\frac{1}{k_o}\frac{g\overline{\Gamma}}{4\pi}B_1 \tag{2.11}$$

$$\overline{w}_{ti} = -\frac{1}{r}\frac{g\overline{\Gamma}}{4\pi}B_2 \tag{2.12}$$

$$\overline{w}_{te} = \frac{1}{r}\frac{g\overline{\Gamma}}{4\pi}\left(1+B_1\right) \tag{2.13}$$

where:
$$B_{1,2} = \left(\frac{1+y_o^2}{1+y^2}\right)^{.25}\left[\frac{1}{e^{gA_{1,2}}-1} \mp \frac{1}{2g}\frac{y_o^2}{\left(1+y_o^2\right)^{1.5}}\ln\left(1+\frac{1}{e^{gA_{1,2}}-1}\right)\right] \tag{2.14}$$

$$A_{1,2} = \pm\left(\sqrt{1+y^2}-\sqrt{1+y_o^2}\right) \mp \frac{1}{2}\ln\left(\frac{\left(\sqrt{1+y_o^2}-1\right)\left(\sqrt{1+y^2}+1\right)}{\left(\sqrt{1+y_o^2}+1\right)\left(\sqrt{1+y^2}-1\right)}\right) \tag{2.15}$$

$$y_o = \frac{r_o}{k_o} = \frac{1}{\tan\left(\beta_{io}\right)} \tag{2.16}$$

$$y = \frac{r}{k_o} = \frac{x}{x_o\tan\left(\beta_{io}\right)} \tag{2.17}$$

Unfortunately, numerical accuracy of the induced velocities is degraded when the reference radius approaches the vortex radius because the expressions go to infinity. This problem can be avoided by using the induction factors from (9) as defined in equations (2.18) and (2.19). The induction factors are the induced velocity components non-dimensionalized by the velocity induced at r by a straight potential vortex extending from z equals zero to positive infinity of strength $\overline{\Gamma}$ situated at r_o. Both the induced velocities and the non-dimensionalizing velocity become infinite in the same order as r approaches r_o. Therefore, the induction factors remain finite as the radii approach each other. Combining the induced velocity equations into the induction factor equations yields the final induction factors as shown in equations (2.20) through (2.23).

$$i_a = \frac{\frac{\overline{w}_a}{\overline{\Gamma}}}{4\pi\left(r-r_o\right)} \tag{2.18}$$

$$i_t = \frac{\frac{\overline{w}_t}{\overline{\Gamma}}}{4\pi\left(r-r_o\right)} \tag{2.19}$$

$$i_{ai} = g\frac{x}{x_o\tan\left(\beta_{io}\right)}\left(\frac{x_o}{x}-1\right)\left(1+B_2\right) \tag{2.20}$$

$$i_{ae} = -g\frac{x}{x_o \tan(\beta_{io})}\left(\frac{x_o}{x} - 1\right)B_1 \tag{2.21}$$

$$i_{ti} = g\left(\frac{x_o}{x} - 1\right)B_2 \tag{2.22}$$

$$i_{te} = -g\left(\frac{x_o}{x} - 1\right)(1 + B_1) \tag{2.23}$$

It is interesting to note that the induction factors are functions of geometric properties (i.e. the relative position of the reference point and where the vortex is shed as well as the angle at which it is shed) and are not dependant on the circulation. Also of note is that the above expressions apply to only one lifting line. In general, there will also be induced velocities at each lifting line (i.e. each blade) from all other lifting lines. However, if the propeller is symmetric, these effects will cancel each other out. (4)

In order to evaluate the accuracy of the assumptions leading to the calculation of the induction factors, Lerbs (4) compared the calculated values to those calculated using direct numerical integration of the Biot-Savart integral by Strscheletzky (10) in the case of g equals 3. He found the method of Strscheletzky produced higher values than the method explained above, but only when i_{ti} is smaller than 0.06. Differences when the induction factor is so small have a negligible effect on the overall final values.

2.2 Velocity Field of Symmetrically Spaced Helical Vortex Sheets

Section 2.1 developed the velocity field for g helical vortex lines. When multiple vortex lines are shed from the same lifting line, a vortex sheet is formed. The velocity components which are produced by the vortex sheets are obtained by integrating the effects of the vortex lines over the span of the lifting line. By integrating over the free vortices and using equations (2.3), (2.18), and (2.19), expressions for the induced velocities (non-dimensionalized with the speed of advance) at station x of the lifting line are shown in equations (2.24) and (2.25). These equations are in terms of the non-dimensional circulation (G) as defined in equation (2.26). (4)

$$\frac{w_a}{V_S} = \frac{1}{2}\int_{x_h}^{1} \frac{dG}{dx_o}\frac{1}{(x - x_o)} i_a dx_o \tag{2.24}$$

$$\frac{w_t}{V_S} = \frac{1}{2}\int_{x_h}^{1} \frac{dG}{dx_o}\frac{1}{(x - x_o)} i_t dx_o \tag{2.25}$$

$$G = \frac{\Gamma}{\pi D V_S} \tag{2.26}$$

Expanding on the method used by Glauert (12) in airfoil theory, the values of the integrals in equations (2.24) and (2.25) can be determined. This requires a new variable, φ, as defined in equation (2.27) which is zero when x equals the hub radius, x_h, and π when x is at the blade tip.

$$x = \frac{1}{2}(1 + x_h) - \frac{1}{2}(1 - x_h)\cos(\varphi) \tag{2.27}$$

For the purpose of analysis, G is estimated to go to zero at the hub and at the tip. For a real propeller, the hub does carry some circulation as described in (12). Therefore, this assumption is not always valid, but it does provide a good first order approximation. This assumption also does not hold true for propellers with a zero-gap duct, but that is beyond the scope of this thesis. Circulation is continuous along the span of the blade, therefore G can be written as the Fourier series shown in equation (2.28). The continuity of G also ensures that single vortices of finite strength do not occur and that the series for G, when placed into equations (2.24) and (2.25), gives the complete induced velocity components. (4)

$$G = \sum_{m=1} G_m \sin(m\varphi) \tag{2.28}$$

In addition to the pitch angle and the number of blades, the induction factors from Section 2.1 also depend on φ and φ_o. The induction factors, with respect to φ_o, can be written as an even Fourier series as described by Schubert (13). This is shown in equation (2.29).

$$i(\varphi, \varphi_o) = \sum_{n=0} I_n(\varphi)\cos(n\varphi_o) \tag{2.29}$$

It follows that the expressions for the non-dimensional induced velocities become as shown in equations (2.30) and (2.31) where h_m^t is defined in equation (2.32). (h_m^a will be defined later.)

$$\frac{w_a}{V_S} = \frac{1}{1 - x_h}\sum_{m=1} m G_m h_m^a(\varphi) \tag{2.30}$$

$$\frac{w_t}{V_S} = \frac{1}{1-x_h}\sum_{m=1} mG_m h_m^t(\varphi) \tag{2.31}$$

$$\begin{aligned} h_m^t(\varphi) &= \int_0^{\pi} \frac{i_t(\varphi,\varphi_o)\cos(m\varphi_o)}{\cos(\varphi_o)-\cos(\varphi)}d\varphi_o \\ &= \frac{1}{2}\sum I_n^t(\varphi)\left\{\int_0^{\pi}\frac{\cos((m+n)\varphi_o)}{\cos(\varphi_o)-\cos(\varphi)}d\varphi_o + \int_0^{\pi}\frac{\cos((m-n)\varphi_o)}{\cos(\varphi_o)-\cos(\varphi)}d\varphi_o\right\} \end{aligned} \tag{2.32}$$

The two integrals inside the summation of equation (2.32) can be solved by following Glauert's method (12). For values of m greater than n, the summation becomes equation (2.33), and for values of m less than n, the summation becomes equation (2.34). With these substitutions made, the final form of h_m^t becomes equation (2.35). A similar process is followed for h_m^a, resulting in equation (2.36).

$$\frac{\pi}{\sin(\varphi)}\sin(m\varphi)\sum_{n=0}^{m} I_n^t(\varphi)\cos(n\varphi) \tag{2.33}$$

$$\frac{\pi}{\sin(\varphi)}\cos(m\varphi)\sum_{n=m+1} I_n^t(\varphi)\sin(n\varphi) \tag{2.34}$$

$$h_m^t(\varphi) = \frac{\pi}{\sin(\varphi)}\left[\sin(m\varphi)\sum_{n=0}^{m} I_n^t(\varphi)\cos(n\varphi) + \cos(m\varphi)\sum_{n=m+1} I_n^t(\varphi)\sin(n\varphi)\right] \tag{2.35}$$

$$h_m^a(\varphi) = \frac{\pi}{\sin(\varphi)}\left[\sin(m\varphi)\sum_{n=0}^{m} I_n^a(\varphi)\cos(n\varphi) + \cos(m\varphi)\sum_{n=m+1} I_n^a(\varphi)\sin(n\varphi)\right] \tag{2.36}$$

Equations (2.35) and (2.36) become indefinite at the hub (φ = 0º) and at the blade tip (φ = 180º). L'Hospital's rule provides values for the functions at these points as shown in equations (2.37) and (2.38).

$$h_m^{t,a}(0°) = \pi\left[m\sum_{n=0}^{m} I_n^{t,a}(0°) + \sum_{n=m+1} nI_n^{t,a}(0°)\right] \tag{2.37}$$

$$h_m^{t,a}(180°) = -\pi\cos(m\cdot 180°)\left[m\sum_{n=0}^{m} I_n^{t,a}\cos(n\cdot 180°) + \sum_{n=m+1} nI_n^{t,a}\cos(n\cdot 180°)\right] \tag{2.38}$$

The above equations allow the induced velocity components to be related to the circulation distribution and the induction factors. The induction factors are known from equations (2.20) through (2.23). Therefore the induced velocity for any circulation distribution that can be represented by equation (2.28) can be calculated, or vice versa. However, it will be necessary to use successive iterations and approximations since the induction factors depend on the pitch angles of the sheets which, in turn, depend on the induced velocities. (4)

2.3 Application to Moderately Loaded Free-Running Non-Optimum Propellers

The vortex sheets described in Section 2.2 are made up of cylindrical vortex lines whose pitch does not change in the axial direction. The next step is to determine the conditions imposed on the variations of the pitch of the vortex lines in the radial direction such that the entire system accurately models the vortex system of an actual propeller.

The force and the flow generated at a lifting line by the vortex sheets of Section 2.2 can be related using an energy balance of the propeller flow. The input power equals the useful power plus the increase in kinetic energy within the volume that passes through the slipstream in a single unit of time. Input power is also made up of an external input plus the work done by the centrifugal pressure on the volume of water in a single unit of time. As stated earlier, for a moderately loaded propeller, it is acceptable to neglect the effects of the centrifugal pressure. Therefore, the energy balance takes the form of equation (2.39) where the integral is taken over a disc with the same diameter as the propeller and ΔE is the kinetic energy of the induced flow within the volume of fluid that passes a point in the ultimate wake in one unit of time. (4)

$$\int^{A} \left(r\omega dQ - v dT \right) = \Delta E \qquad (2.39)$$

At the lifting line, the left hand side of equation (2.39) can be expressed in terms of the relative flow's pitch by substituting dQ and dT with their respective expressions from the Kutta-Joukowski law. The Kutta-Joukowski law states that the force at the lifting line is perpendicular to the resultant incoming flow. Using the nomenclature of Figure 2, the equivalent of the left hand side is shown in equation (2.40). (4)

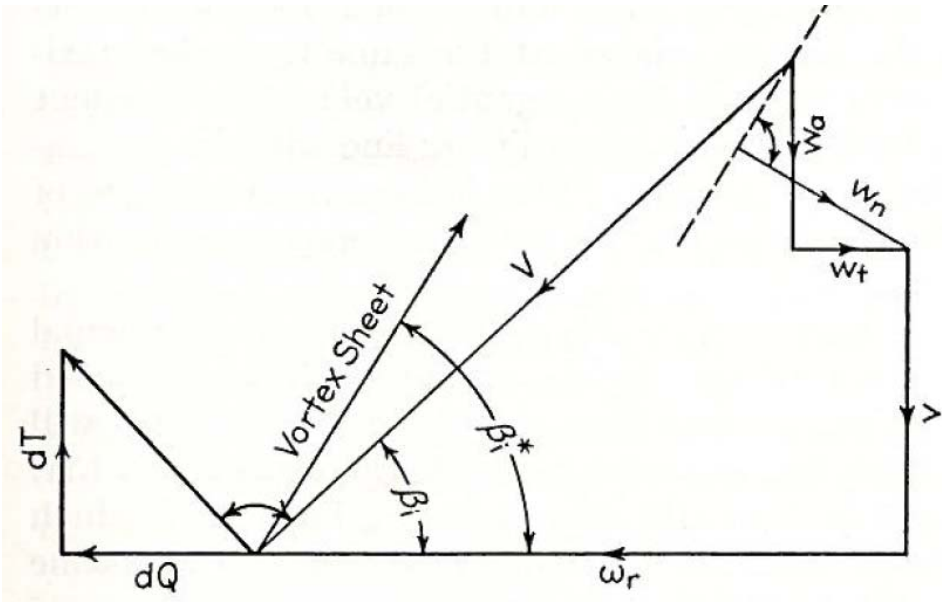


Figure 2: Pitch Angle Relationships to Velocities and Forces (4)

$$\int^{A}\left(r\omega dQ - v dT\right) = \rho\int_{r_h}^{R}\left(g\Gamma\right)\left(v + w_a\right)\left(\frac{w_a}{\tan\left(\beta_i\right)} + w_t\right)dr \qquad (2.40)$$

Using Green's theorem, the kinetic energy of the flow which is included in a certain volume can be represented by an integral over the surface of the volume. Applying this to the right hand side of equation (2.39) allows it to be expressed in terms of the pitch of the vortex lines as shown in equation (2.41). (4)

$$\begin{aligned} E &= \rho\int_{r_h}^{R}\left(g\Gamma\right)\left(v + w_a\right)\frac{w_n}{\sin\left(\beta_i^*\right)}dr \\ &= \rho\int_{r_h}^{R}\left(g\Gamma\right)\left(v + w_a\right)\left[\frac{w_a}{\tan\left(\beta_i^*\right)} + w_t\right]dr \end{aligned} \qquad (2.41)$$

Since the left hand side must equal the right hand side, the integrals in equations (2.40) and (2.41) must be the same. Therefore $\beta_i(r)$ must equal $\beta_i^*(r)$. In other words, the vortex lines must leave the lifting line at a pitch angle equal to the angle of the resultant inflow. The angle of the resultant flow is dependent on the accuracy of the approximations of the induced flow distributions, $w_a(x)$ and $w_t(x)$. The level of accuracy of the flows caused by the vortex sheets in Section 2.2 is ascertained by comparing them to a known solution of the flow past a propeller.

One solution for comparison is that of an optimum propeller for which the Betz condition, defined in equation (2.42), holds. (4)

$$\frac{v}{\omega r}\frac{\omega r - 2w_t}{v + 2w_a} = \text{constant} \tag{2.42}$$

Equation (2.43) shows the expansion of both the numerator and denominator of the Betz condition. This shows that the Betz condition is met if the induced velocity terms of order two or greater can be neglected. From this it is expected that for an arbitrary circulation distribution, the induced velocity at a lifting line is sufficiently estimated from the vortex sheets of Section 2.2 when higher order terms of the axial and tangential induced velocities are small compared to v and ωr, respectively. To this degree, the difference in the shape of the actual and modeled vortex sheets is insignificant. (4)

$$\begin{aligned} \omega r - 2w_t &= (\omega r - w_t)\left(1 - \frac{w_t}{\omega r} - \frac{w_t^2}{\omega^2 r^2} - \ldots\right) \\ v + 2w_a &= (v + w_a)\left(1 + \frac{w_a}{v} - \frac{w_a^2}{v^2} + \ldots\right) \end{aligned} \tag{2.43}$$

2.4 Determining the Circulation and Induced Velocities

Given that the geometry and polar curves of a propeller are known, it is possible to estimate the circulation and induced velocity distributions for a given advance ratio. In order to ease the calculations, the advance coefficient will be used in lieu of the advance ratio. The advance coefficient is defined as the advance ratio divided by pi.

The first step is to use the Kutta-Joukowski law and the nomenclature of Figure 3 from (4) to derive the relationship between the lift coefficient and the circulation at any radius along the blade as shown in equation (2.44). The solidity of the propeller, s, is defined in equation (2.45) where l is the chord length at a specified radius.

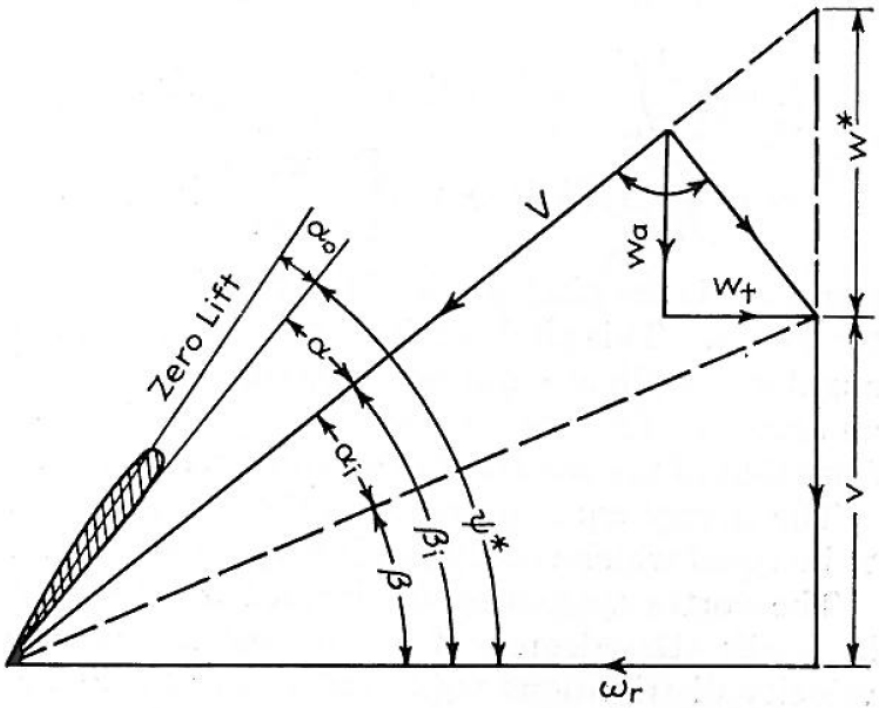


Figure 3: Velocity Diagram of a Moderately Loaded Propeller (4)

$$C_L s = \frac{2gGv}{V} \tag{2.44}$$

$$s = \frac{gl}{\pi D} \tag{2.45}$$

For a small angle of attack α, the lift curve slope is approximately constant. Therefore, the lift coefficient can be found as shown in equation (2.46) which also shows the rearrangement of the different angular values base on Figure 3. Figure 3 can also be used to determine expressions for both the non-dimensional inflow velocity and the tangent of ($\beta+\alpha_i$) as functions of the advance coefficient which are shown in equations (2.47) and (2.48), respectively.

$$C_L = \frac{dC_L}{d\alpha}(\alpha + \alpha_o) = \frac{dC_L}{d\alpha}\left[(\psi^* + \alpha_o) - (\beta + \alpha_i)\right] \tag{2.46}$$

$$\frac{V}{v} = \frac{\frac{x}{\lambda} - \frac{w_t}{v}}{\cos(\beta + \alpha_i)} \tag{2.47}$$

$$\tan(\beta+\alpha_i)=\frac{\left(1+\frac{w_a}{v}\right)}{\left(\frac{x}{\lambda}-\frac{w_t}{v}\right)} \tag{2.48}$$

Combining equations (2.44), (2.46), and (2.47) results in equation (2.49).

$$\left(\frac{x}{\lambda}-\frac{w_t}{v}\right)\left[s\frac{dC_L}{d\alpha}\left(\psi^*+\alpha_o-\beta\right)-s\frac{dC_L}{d\alpha}\left(\alpha_i\right)\right]=2gG\cos\left(\beta+\alpha_i\right) \tag{2.49}$$

The next step is to substitute the expressions for w_a, w_t, and G (defined above in equations (2.24), (2.25), and (2.28)) into equations (2.49) and (2.48). The results of this substitution are shown in equations (2.50) and (2.51).

$$\frac{dC_L}{d\alpha}s\frac{x}{\lambda}\left[\left(\psi^*+\alpha_o-\beta\right)-\alpha_i\right]=\ldots$$
$$\sum_{m=1}G_m\left\{2g\sin(m\varphi)\cos\left(\beta+\alpha_i\right)+\frac{mh_m^t}{1-x_h}\frac{dC_L}{d\alpha}s\left[\left(\psi^*+\alpha_o-\beta\right)-\alpha_i\right]\right\} \tag{2.50}$$

$$\tan(\beta+\alpha_i)=\frac{1+\frac{1}{1-x_h}\sum_{m=1}mG_mh_m^a}{\frac{x}{\lambda}-\frac{1}{1-x_h}\sum_{m=1}mG_mh_m^t} \tag{2.51}$$

The final step in finding an approximation for the circulation distribution and the induced velocities at the lifting line is to satisfy equations (2.50) and (2.51) at m stations along the blade. Successive iterations become necessary by starting with a reasonable guess for the distribution of α_i and solving equation (2.50) for the Fourier coefficients, G_m. These coefficients are then used in equation (2.51) to determine a new α_i distribution. The average of the new and the old α_i distributions are taken as the starting distribution for the next round of iteration. This process is repeated until the new and the old α_i distributions differ by only a small amount. A difference of less than $\pm 0.2°$ is considered sufficient accuracy. (4)

3. Implementation and Validation

Given the basic geometry and polar curves of a propeller, along with an educated guess for the hydrodynamic pitch angle of the resultant inflow, the theories of Section 2 will produce the circulation and induced velocity distributions at the lifting line approximation of the propeller blade for a given advance ratio. This data can then be used with the *Forces.m* function defined in OpenProp to determine the thrust, torque, and efficiency information for the propeller.

3.1 MATLAB® Implementation of the "Building Blocks"

In order to implement the theories of Section 2 as a fast and reliable propeller analysis tool, the equations were coded using MATLAB®. Although the only equations that are necessary for the analysis are equations (2.50) and (2.51) , the major required inputs were coded as separate functions in order to provide greater understanding to the reader as well as greater ease in troubleshooting the code. All of the MATLAB® functions described in Section 3 can be found in Appendix A.

The first function in the process is named *Find_Lerbs_Induction_Factors.m*. This function takes as input the non-dimensional radial coordinate of a control point and a vortex shedding point in addition to the number of blades of the propeller and the pitch distribution of the vortex sheets. Then, using equations (2.20) through (2.23), it calculates the axial and tangential induction factors. The function can only do this process for a single set of inputs, therefore it must be called multiple times in a loop to determine the induction factors at all control points caused by the entire vortex system.

The next function is named *Calculate_Induction_Fourier_Coefficients.m*. This function also takes as input the non-dimensional radial coordinates of control points and vortex shedding points with the axial and tangential induction factors determined in the previous function. It returns both the axial and tangential Fourier coefficients defined in equation (2.29).

The next function is named *Calculate_hm_factors.m*. This function takes the induction Fourier coefficients and the non-dimensional radial coordinates of the control points as inputs in order to

return the factors h_m^a and h_m^t from the induced velocity equations as they are defined in equations (2.35) through (2.38).

3.2 Validation of the "Building Blocks" with Lerbs' Example Data

The functions in Section 3.1 provide the basic building blocks needed to solve for the unknown circulation distribution and induced velocities. In order to ensure that the MATLAB® functions did not contain errors, their outputs were compared to the data from the example in Part 2 Section A-1 of the Lerbs paper (4). The example consists of a 4-bladed propeller with a zero hub radius at an advance coefficient of 0.2. The specific section data is shown in Table 1.

$\varphi\,(^\circ)$	x	$\beta_{io}\,(^\circ)$
0	0.000	90.00
30	0.067	75.55
60	0.250	46.12
90	0.500	27.47
120	0.750	19.12
150	0.933	15.58
180	1.000	14.57

Table 1: Input Data for Lerbs' Example

Table 2 and Table 3 show the difference in the axial and tangential induction factors calculated by the *Find_Lerbs_Induction_Factors.m* function and those presented by Lerbs in (4). The minor differences are expected since Lerbs read the induction factors off of a graph and the function calculates them from equations.

Radial Location of Shed Vortices (deg)	Radial Location of Control Points (deg)				
	30	60	90	120	150
0	0.000	0.000	0.000	0.000	0.000
30	0.000	-0.006	0.000	0.000	0.000
60	-0.006	0.000	-0.027	0.000	0.000
90	0.003	0.000	0.000	-0.050	-0.009
120	-0.002	0.004	-0.010	0.000	-0.140
150	-0.004	-0.018	0.015	0.021	0.000
180	-0.002	0.022	-0.022	0.000	-0.008

Table 2: Axial Induction Factor Difference

Radial Location of Shed Vortices (deg)	Radial Location of Control Points (deg)				
	30	60	90	120	150
0	0.000	0.000	0.000	0.000	0.000
30	0.000	0.006	0.002	0.001	0.001
60	0.000	0.000	0.005	0.001	0.002
90	0.000	-0.002	0.000	0.012	-0.001
120	0.000	0.000	-0.002	0.000	0.004
150	0.000	0.000	-0.008	-0.011	0.000
180	0.000	0.000	-0.004	-0.009	-0.005

Table 3: Tangential Induction Factor Difference

Table 4 and Table 5 show the difference in the axial and tangential induction factors Fourier coefficients calculated by the *Calculate_Induction_Fourier_Coefficients.m* function and those presented by Lerbs in (4). The occasional minor differences are in the third decimal place which is the extent of the accuracy of the data in (4).

Radial Location of Control Points (deg)	Induction Factor Fourier Coefficients						
	0	1	2	3	4	5	6
30	0.000	0.000	0.000	0.000	0.000	0.001	0.000
60	0.000	0.000	0.000	0.000	0.000	0.000	0.001
90	0.000	0.000	0.000	0.000	0.001	0.000	0.000
120	0.000	0.000	0.000	0.000	0.000	0.000	0.000
150	-0.001	0.000	-0.001	0.000	0.001	0.000	0.001

Table 4: Axial Induction Factor Fourier Coefficient Difference

Radial Location of Control Points (deg)	Induction Factor Fourier Coefficients						
	0	1	2	3	4	5	6
30	0.000	0.000	0.001	0.000	0.001	-0.001	0.000
60	0.000	0.000	0.000	0.000	0.000	0.000	0.000
90	0.000	0.000	0.000	0.000	0.000	0.000	0.000
120	0.000	0.000	0.000	0.000	0.000	0.000	0.000
150	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 5: Tangential Induction Factor Fourier Coefficient Difference

Table 6 and Table 7 show the difference in the factors h_m^a and h_m^t from the induced velocity equations as calculated by the *Calculate_hm_factors.m* function and those presented by Lerbs in (4). The occasional minor differences are typically in the third decimal place which is the extent of the accuracy of the data in (4).

Radial Location of Control Points (deg)	Induction Factor Fourier Coefficients				
	1	2	3	4	5
30	-0.004	-0.006	-0.002	0.001	-0.001
60	0.000	0.002	0.000	-0.001	-0.005
90	0.000	0.000	0.001	0.000	0.000
120	-0.002	0.001	-0.002	0.000	-0.003
150	-0.002	0.004	-0.006	0.000	-0.002

Table 6: Axial h_m Factor Difference

Radial Location of Control Points (deg)	Induction Factor Fourier Coefficients				
	1	2	3	4	5
30	-0.002	-0.006	-0.002	0.003	-0.005
60	-0.001	-0.003	0.000	0.003	-0.001
90	-0.001	0.000	0.000	0.000	0.000
120	-0.001	0.000	-0.002	0.001	-0.003
150	-0.011	0.002	-0.001	-0.001	0.001

Table 7: Tangential h_m Factor Difference

The above results show that the "building block" functions have sufficient accuracy to be used in the calculation of circulation and induced velocities.

3.3 MATLAB® Implementation for Finding Circulation, Induced Velocities, and the Resultant Forces

With the building blocks in place, the next step is to use them to solve equations (2.50) and (2.51). This is done using the *Calculate_Gm_and_new_AlphaI.m* function. This function returns estimations of the Fourier coefficients for the non-dimensional circulation function shown in equation (2.28), the unknown angle α_i from equation (2.51), and the components of the induced velocity from equations (2.30) and (2.31) all of which are based on the initial estimate of α_i. The inputs required by the function are grouped into two main categories, the propeller geometry and its operating conditions.

The important aspects of propeller geometry include the solidity of the propeller as defined in equation (2.45) as well as the pitch angle and the foil section data at specified non-dimensional radial coordinates. The first required value of foil data is the slope of the lift curve as a function of angle of attack. Appendix IV of Abbott (2) provides the polar plots of many NACA 4, 5, 6, and 7 series airfoils. If the propeller blade is made up of one of these foils, then the slope of the

curve can be determined graphically. However, this is not convenient when automating the process with MATLAB®. Wing theory states that the theoretical slope of the lift curve is 2π radians (2). Experimental results have shown close agreement with this value, therefore it will be used for this thesis.

The next required piece of data for the foil sections is the angle of attack that produces no lift (also called the angle of zero lift). Abbott (2) provides many ways to determine this angle. If the propeller blade is made up of foil section contained in Appendix IV of (2), the angle of zero lift can be graphically determined from the polar plots. Again, this is not convenient for the computing environment. If the mean line distribution is known, Abbott outlines a method derived by Munk to estimate the angle of zero lift with equation (3.1). It requires that the ordinates of the mean line as a percent of the chord, y_1 through y_5, are known at specific chord-wise percentages, x_1 through x_5, defined below. The constants k_1 through k_5 are defined such that the angle of zero lift is returned in degrees.

$$-\alpha_o = k_1 y_1 + k_2 y_2 + k_3 y_3 + k_4 y_4 + k_5 y_5 \qquad (3.1)$$

$$\begin{array}{rlll} & k_1 = 1252.24 & & x_1 = 0.99458 \\ & k_2 = 109.048 & & x_2 = 0.87426 \\ \textit{where}: & k_3 = 32.5959 & \textit{and} & x_3 = 0.50000 \\ & k_4 = 15.6838 & & x_4 = 0.12574 \\ & k_5 = 5.97817 & & x_5 = 0.00542 \end{array}$$

The function *Calculate_Angle_of_Zero_Lift.m* approximates the angle of zero lift for a NACA a=0.8 mean line foil. The function is easily adaptable to other mean lines by entering new x-y coordinates. The coordinates of many mean lines are listed in Abbott's Appendix II (2). The function takes as input the maximum chamber ratio and scales the mean line ordinates from the design conditions to the desired conditions. It then uses equation (3.1) to calculate and return the angle of zero lift in degrees.

The final inputs for the *Calculate_Gm_and_new_AlphaI.m* function are the operating conditions. They consist of the advance coefficient, angle of flow, the factors h_m^a and h_m^t, and the estimation

of α_i. The propeller's rotational and axial speeds dictate the advance coefficient (as described in Section 2.4) as well as the angle of flow, β, as shown in Figure 3. The factors h_m^a and h_m^t are defined in equations (2.35) and (2.36). The initial guess of the angle α_i can be obtained from OpenProp.

With all of the inputs determined, the *Calculate_Gm_and_new_AlphaI.m* function starts with equation (2.50). The left-hand side of the equation is calculated for every control point and the values are stored in a row vector of k elements, where k is the number of control points. Next, everything inside the curly brackets on the right-hand side of the equation is calculated at all k control points using the values 1 to k for m thus creating a k by k matrix. The complete right-hand side of equation (2.50) is the sum of the non-dimensional circulation Fourier coefficients (G_m) times everything inside the curly brackets. In matrix notation the sum of the products is simply the matrix multiplication of the curly bracket matrix (k by k) with G_m (k by 1). The built-in MATLAB® function *linsolve* is then used to solve for the unknown G_m column vector.

Now that the coefficients for the circulation series (based on the estimate of α_i) are known, equation (2.51) is used to calculate a new estimate for the distributions of α_i and subsequently β_i. Next, β_i is used as the vortex sheet pitch angle distribution. This leads to new induction factors with their own Fourier coefficients. These new coefficients are used to calculate new h_m^a and h_m^t factors for input again into the *Calculate_Gm_and_new_AlphaI.m* function. The values of the circulation series coefficients are also used to estimate the axial and tangential velocities in equations (2.30) and (2.31). This process is repeated until the original and new estimate of α_i agree to within $\pm 0.2°$ or a predetermined number of iterations are performed.

Once there is convergence on the value of α_i, the *Forces.m* function from OpenProp is used. This function is shown in Appendix A for completeness, but it was not altered from its most recent release. The forces of note for propeller analysis, as explained in Section 1, are the thrust coefficient, the torque coefficient, and the efficiency.

3.4 Validation of Circulation, Induced Velocity, and Forces Using OpenProp

OpenProp was chosen to design a propeller because it has been proven to provide accurate circulation and induced velocity distributions for a given design point. OpenProp also designs the blade shape and pitch angle based on wing theory. Therefore it should be possible to use all of the geometric data and the Lerbs method (4) to arrive back at the same circulation and induced velocity distributions. Lerbs modeling of the circulation as a Fourier sine series necessitates that the circulation goes to zero at both the hub and blade tip. For this reason, in designing the propeller with the Single Propeller Design option in OpenProp, the "Hub Image Flag" and "Ducted propeller" options were both left unchecked. Throughout the validation process, errors were discovered within the OpenProp code that affected the way the blades were designed but not the calculation of the circulation and induced velocity distributions.

The first error noted was in the calculation of "Vstar" just prior to generating the "OpenProp_Performance.txt" reporting file. This value was supposed to be the magnitude of the resultant inflow to the lifting line at each control point non-dimensionalized by the speed of the ship. However, the "ωr" term had units of length over time. This problem was solved by dividing the term by the speed of the ship. This error also cascaded into the circulation, "Gamma", and the lift coefficient, "Cl", reported in "OpenProp_Performance.txt" file. This lift coefficient was used by the *Geometry.m* function to scale the chamber of the mean line.

The next error detected also affects the scaling of the chamber of the mean line. The Single Propeller Design graphical user interface allows the designer to enter the maximum chamber divided by chord length for multiple radial positions along the blade. This information in turn is multiplied by the lift coefficient in the *Geometry.m* function to set a new maximum chamber ratio. According to Abbott (2), the maximum chamber ratio can be scaled by a constant factor to produce the desired lift; however, the factor must be as shown in equation (3.2). In general, if the designer inputs an arbitrary maximum chamber distribution, the corresponding lift coefficient is unknown and therefore the scaling factor cannot be determined. This problem was solved for this validation run by inputting the maximum chamber ratio for the NACA a=0.8 mean line from Appendix II of (2) into the graphical user interface. This data corresponds to a lift coefficient of 1.0. Thus the scaling factor is simply the desired lift coefficient.

$$Scaling\ Factor = \left(\frac{C_{L\ Desired}}{C_{L\ Original}} \right) \tag{3.2}$$

The final error detected was with the calculation of the pitch angle in OpenProp. The program simply took the "Ideal Angle of Attack" inputted in the graphical user interface and added it to the hydrodynamic pitch angle, β_i, that was calculated as part of the optimization routine. As described in Abbott (2), if the chamber is scaled by a constant factor to achieve a desired lift, the ideal angle of attack must also be scaled by the same factor. This problem was solved by multiplying the ideal angle of attack by the factor in equation (3.2) before determining the pitch angle of the foil section.

OpenProp was used to design a single propeller using all of the default settings except for the modifications noted above. The outputs of that design process were then used as inputs to the Lerbs method for determining the circulation and induced velocity distributions. Figure 4 shows both the OpenProp and Lerbs method non-dimensional circulation distributions. Figure 5 show the induced velocity components for both the OpenProp and Lerbs method. Both figures show close agreement in the values. The largest differences occur at the blade root. It is possible this difference occurs because the Lerbs method forces the circulation to be zero at the hub. Even though the option to eliminate hub effects in OpenProp was chosen, the OpenProp circulation does not approach zero as quickly as the Lerbs method.

With accurate estimates of the circulation and velocities, the next step is to calculate the resultant forces and compare them to the forces calculated in OpenProp. The values of the thrust coefficient, the torque coefficient, and the efficiency from using both the Lerbs method and OpenProp are shown in Table 8. The maximum difference between the values is only approximately .1%. It is concluded that the minor differences in the velocity and circulation values have a negligible effect on the forces produced by the propeller.

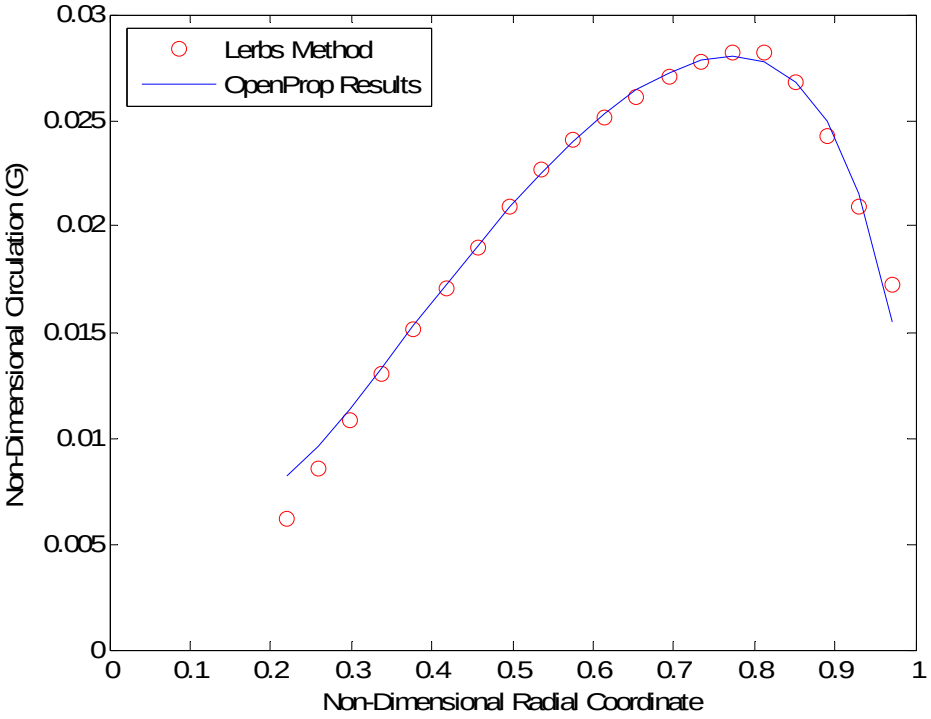


Figure 4: Non-Dimensional Circulation Comparison

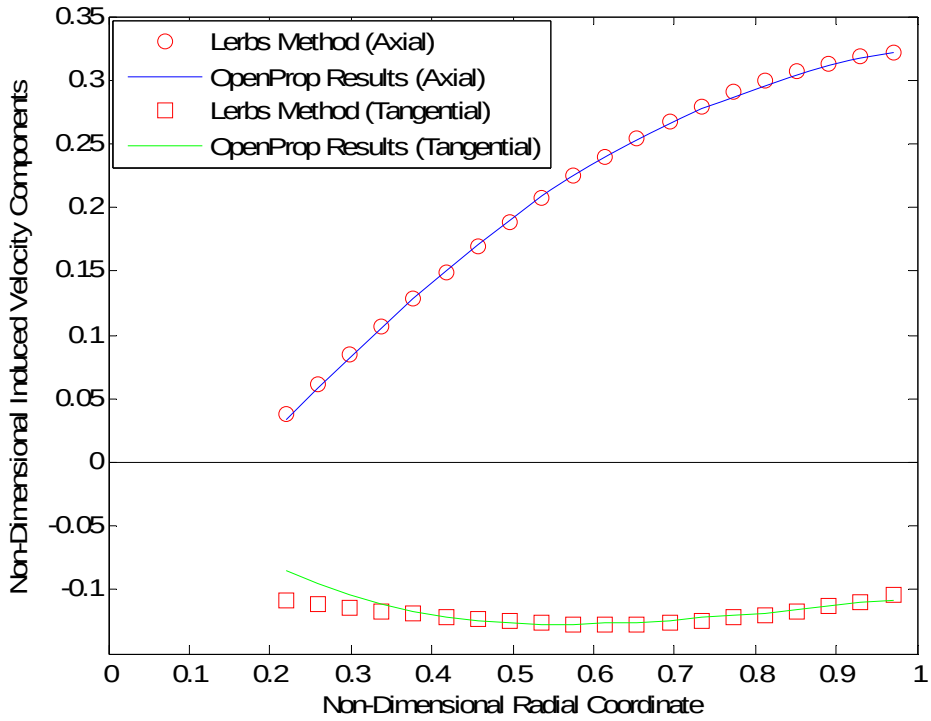


Figure 5: Non-Dimensional Induced Velocities Comparison

	OpenProp Results	Lerbs Method	Percent Difference (%)
K_T	0.220	0.219	0.105
$10*K_Q$	0.377	0.376	0.038
Efficiency	0.696	0.695	0.067

Table 8: Thrust, Torque, and Efficiency Differences

3.5 MATLAB® Implementation for Determining Propeller Performance Characteristics at Off-Design Advance Ratios

The *Some_Open_Water_Characteristics.m* function follows essentially the same process outlined in Section 3.3. The only exception is that instead of only looking at just one advance ratio value, the function loops through the process for a range of advance ratio values. For each advance ratio, the converged values of α_i and β_i are used as the initial estimates for the next advance ratio. The inputs to the function are grouped into the following three main categories: items contained in the "design" structure defined in OpenProp v3.0, items that are defined by the propeller's geometry but are not part of the "design" structure, and the advance ratio range of interest.

Many of the parameters from the "design" structure are defined at a set of control points; therefore, one of the first pieces of information that is required is the non-dimensional radial coordinate of those control points. The chord-over-diameter ratio, the hydrodynamic pitch angle, the (non-induced) non-dimensional velocity components, and the drag of the foil section must all be defined at the control point locations. Other propeller geometry data from the "design" structure that is required includes the hub radius, the number of blades, and the radius of the vortex that is shed from the hub. The last parameters required from the "design" structure are the size of the radial increment used to approximate the force integrations, the volumetric mean inflow velocity, the hub image flag, and the duct thrust coefficient. Although the scope of this thesis does not include ducted propellers, a value of zero thrust must be entered in order to keep the *Forces.m* function unchanged from OpenProp.

The parameters that are not part of the "design" structure but are defined at the control points are the pitch over diameter and the maximum chamber ratio distributions. The diameter of the propeller is also required. The last required input is the advance ratio range over which the propeller is to be analyzed.

The value of α_i is easily obtained from the information contained in the "design" structure. OpenProp provides this information at the design point of the propeller. Therefore, the *Open_Water_Characteristics.m* function starts at the design advance ratio and then works to both extremes of the advance ratio range of interest with a step size of 0.01. The *Open_Water_Characteristics.m* function calls the *Some_Open_Water_Characteristics.m* function first for the range of advance ratios less than the design value and then again for the range above the design value. After calculating the forces and efficiencies over the entire range, the function checks to see if the efficiency ever becomes negative. This point is where the propeller is "wind-milling" and no longer providing any useful thrust to the ship. The function then returns the J_S, K_T, K_Q, and efficiency values for points before the "wind-milling" point.

3.6 Validation of Propeller Performance Characteristics at Off-Design Advance Ratios

The same propeller designed by OpenProp in Section 3.4 was used to validate the propeller performance MATLAB® code described in Section 3.5. The result is shown in Figure 6. In order to determine the robustness of the process described in Section 3.5, the code was altered so that it would not only start at the design point and work toward the endpoints, but to also start at the lowest value in the advance ratio range and work up to the highest and vice versa. In both cases, the initial guess of α_i (corresponding to the design point) was used. The results are shown in Figure 7. There is no significant difference in the results of the three methods; therefore, the method is not very sensitive to the initial guess of α_i.

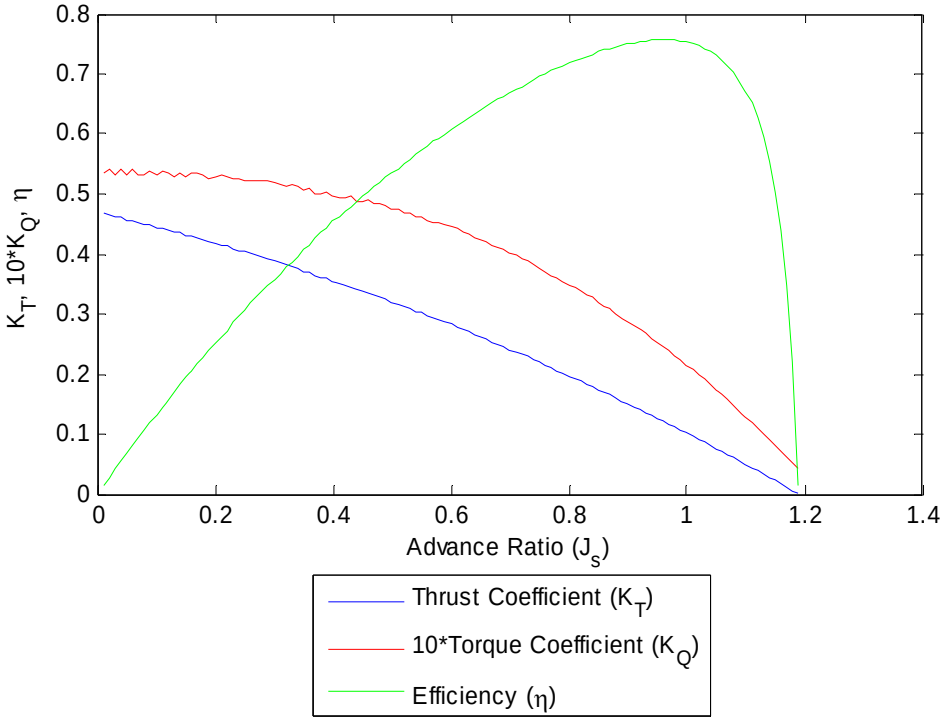


Figure 6: Performance Curves of a Generic Propeller

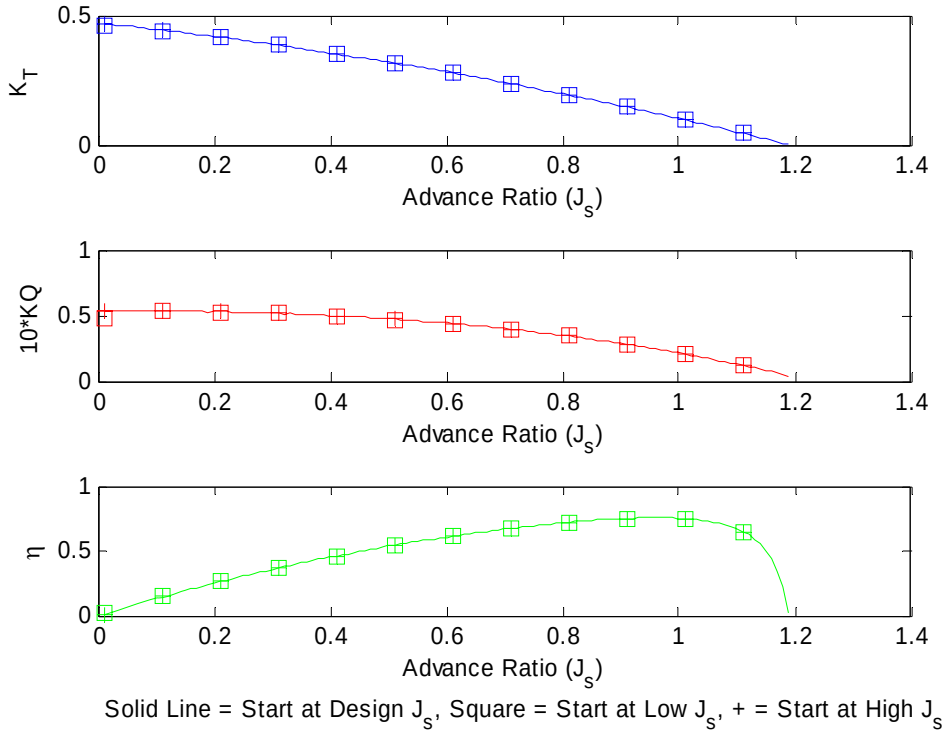


Figure 7: Robustness Results for the Initial Estimation of α_i

It is important to note that the general shape of the curves in Figure 6 is similar to the examples in Woud (1). However, there is no data for this particular propeller to validate that the values are accurate. Therefore, the numerical accuracy of the program is validated using the DTMB 4119 propeller. The DTMB 4119 propeller was chosen for the validation due to its relatively simple geometry (it has no rake or skew) and the extensive amount of experimental data available for it. The geometric characteristics of the DTMB 4119 propeller as reported by Black (14) are shown in Table 9.

Diameter, D = 1.00 ft (.305m)
Rotation: Right Hand
Number of Blades: 3
Hub-Diameter Ratio: 0.20
Skew θ_s, Rake: None
Design Advance Coefficient, J: 0.833
Section Thickness Form: NACA66 (DTMB Modified)
Section Meanline: NACA a=0.8
Design Thrust Coefficient, K_T: 0.150

r/R	c/D	P/D	θ_s	i_T/D	t_M/c	f_M/c
0.2	0.3200	1.105	0	0	0.20550	0.01429
0.3	0.3625	1.102	0	0	0.15530	0.02318
0.4	0.4048	1.098	0	0	0.11800	0.02303
0.5	0.4392	1.093	0	0	0.09016	0.02182
0.6	0.4610	1.088	0	0	0.06960	0.02072
0.7	0.4622	1.084	0	0	0.05418	0.02003
0.8	0.4347	1.081	0	0	0.04206	0.01967
0.9	0.3613	1.079	0	0	0.03321	0.01817
0.95	0.2775	1.077	0	0	0.03228	0.01631
1	0.0000	1.075	0	0	0.03160	0.01175

Table 9: Geometry of the DTMB 4119 Propeller (14)

This data from Table 9 was entered into OpenProp in the Single Propeller Design GUI shown in Figure 8. The propeller produced by OpenProp with these inputs is shown in Figure 10. The thickness form of the DTMB 4119 propeller is the NACA 66 (DTMB modified). OpenProp does not currently design propellers with this thickness form so the NACA 65A010 form was used instead. Neither OpenProp nor any of the MATLAB® functions described above use the thickness of the blade in any calculations. Therefore, the difference in the thickness forms is not

considered a problem for the validation of the performance curves. The propeller produced by OpenProp also has a different pitch over diameter distribution than the actual DTMB 4119 propeller as shown in Figure 9. The actual distribution was inputted into the *Open_Water_Characteristics.m* function for validating the results.

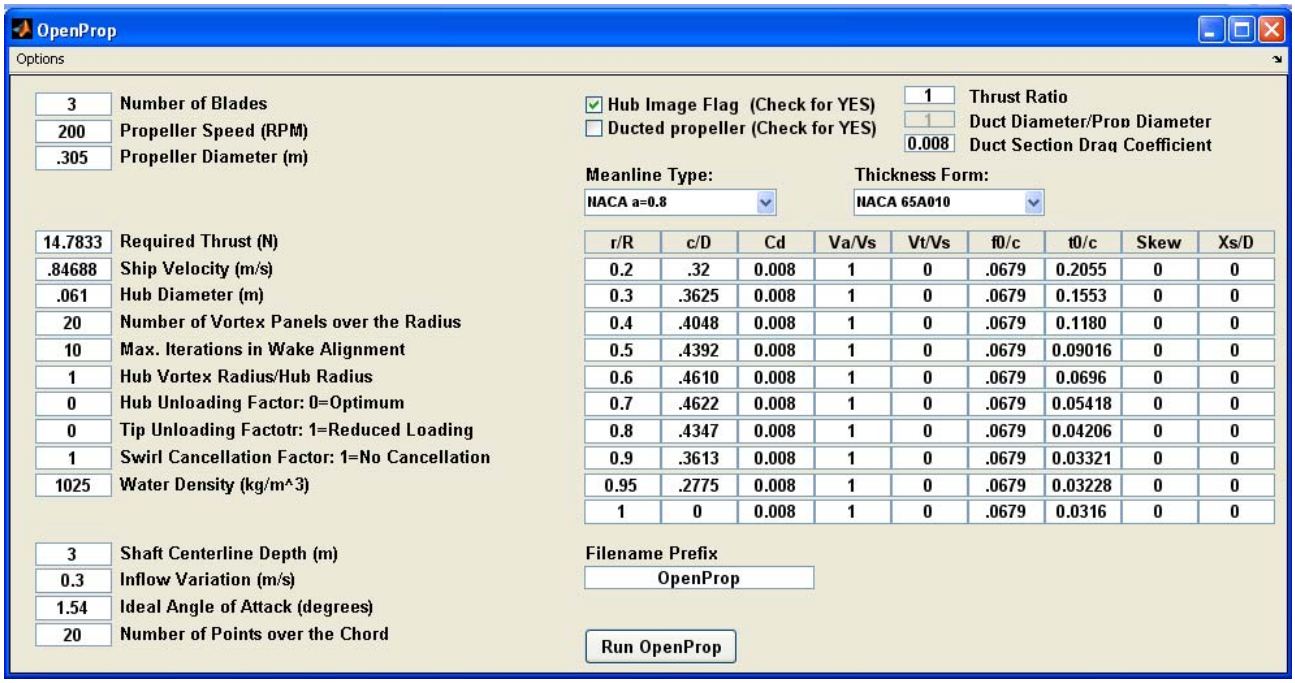


Figure 8: OpenProp Input Parameters for the DTMB 4119 Propeller

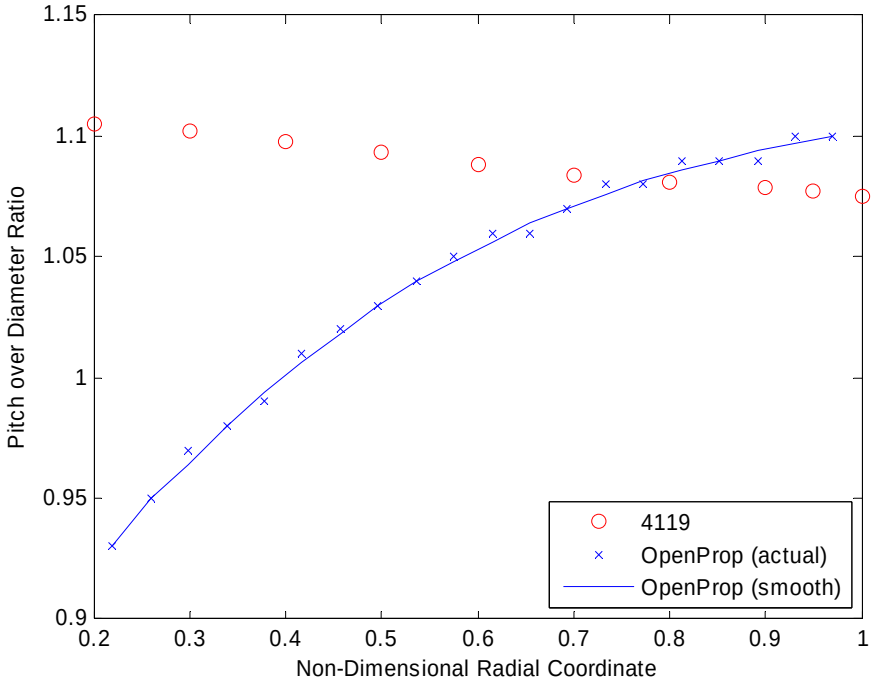


Figure 9: Pitch over Diameter Ratio for the DTMB 4119 Propeller and OpenProp Output

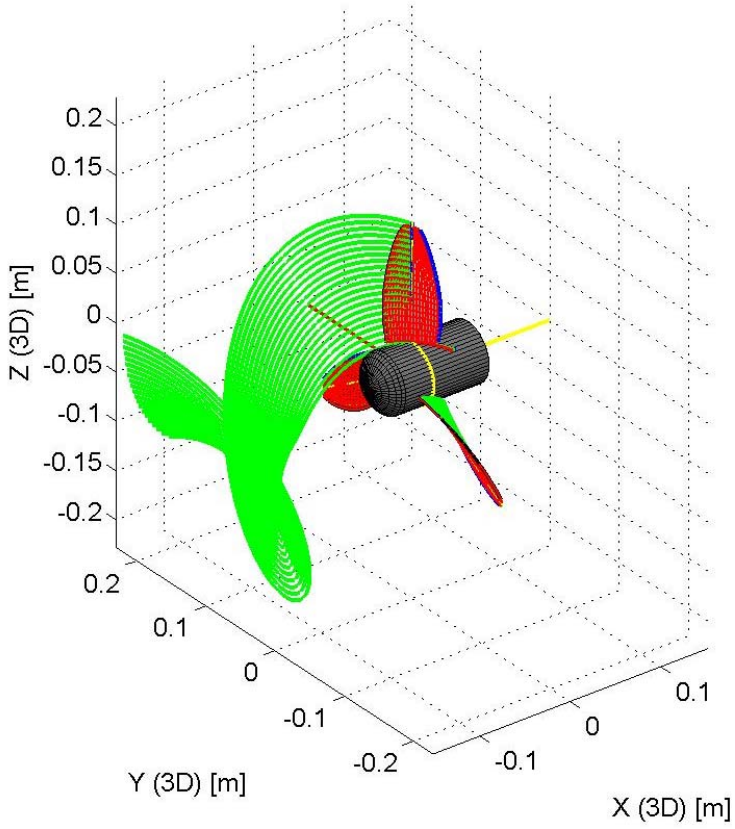


Figure 10: OpenProp Representation of the DTMB 4119 Propeller

The propeller performance results for the DTMB 4119 propeller generated by the MATLAB® code is shown with the solid lines in Figure 11. The X's in Figure 11 represent the experimentally derived performance values as reported by Hsin and Kerwin in (15). The dashed lines in Figure 11 represent the performance values produced by the program MIT-PSF-10 when neglecting the hub effects as reported in (15). The Lerbs method, which also neglects the hub effects, shows close agreement with the MIT-PSF-10 results. The most significant differences occur in the torque coefficient and efficiency values at low and high values of advance ratio. One possible reason for the deviations at low J_S values is that the Lerbs method is only valid for moderately loaded propellers. The load increases with decreases in the advance ratio therefore some error in this region is expected. For higher values of J_S, the propeller is getting closer to the "wind-milling" point discussed in Section 3.5. At the "wind-milling" point the propeller is providing no lift. Recall that the angle of zero lift is estimated using Munk's method. Any deviation between this estimation and the actual angle of zero lift will be seen as the advance ratio nears the "wind-milling" point.

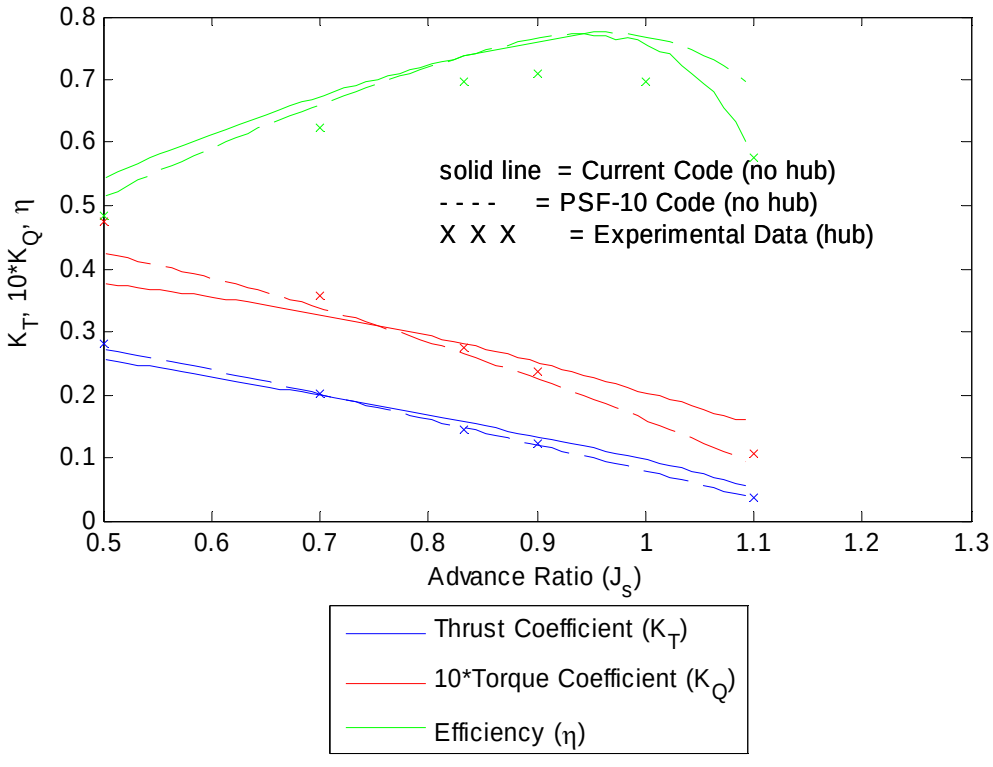


Figure 11: DTMB 4119 Propeller Performance Curves

4. Conclusions and Recommendations

4.1 Conclusions

This thesis successfully implements the methods derived by Lerbs for determining the circulation and induced velocity distributions of a propeller using MATLAB® code. The only requirements that are necessary are that the geometry of the propeller and a reasonable estimate of the angle of the incoming flow are known. Furthermore, the circulation and velocities are used to determine the thrust coefficient, torque coefficient, and the efficiency of the propeller. These values are calculated at multiple advance ratios in order to produce an open water performance diagram for the propeller.

The performance curves show excellent agreement with OpenProp at the design point of the propeller. They also show adequate agreement with experimental and more sophisticated numerical programs both at the design point and at off-design conditions. The code and theories presented in this thesis utilize lifting line theory which is often used at the early stage of propeller design. The MATLAB® code explained above will give the designer propeller performance information at off-design advance coefficients that would normally not be available until more detailed design and modeling was completed. In addition, the performance curves are provided in a matter of one to two minutes. This allows the designer to make changes to the propeller and quickly analyze if those modifications had the desired effect.

4.2 Recommendations for future work

One area of future work that would enhance the features of this thesis is the incorporation of this work into the OpenProp code. The code was designed with this feature in mind; therefore it should be relatively quick to implement. Currently the only road block is that OpenProp does not save the variables for the maximum chamber or pitch over diameter distributions. Since these are both calculated in the *Geometry.m* function, the variables would have to be added as outputs to that function and saved with an appropriate name in the main part of the OpenProp code. The user interface would also have to be updated to give the user the option to produce the

performance curves and the desired advance ratio range. Obviously, additional output reports could also be generated so the performance can be viewed in tabular form as well.

A second area of future work to enhance the features of this thesis is to obtain better estimates of the characteristics of the foil sections. This could be done by building on Peterson's work (7) and have MATLAB® use XFOIL to obtain foil data. Examples of ways XFOIL can be used to improve the accuracy of the model include using it to determine the angle of zero lift, the slope of the lift curve, and detailed viscous data for the foils. XFOIL can also be used to obtain cavitation information at any advance ratio.

Another area of future work involves enhancing the Lerbs method to work when the circulation at the hub and/or tip of the blade does not go to zero. This is needed since the hub typically carries a certain amount of circulation that is shed in a hub vortex. Additionally, if the propeller had a zero gap duct around it, the circulation at the blade tip would be a non-zero value. One possible way to approach this to use a method of images to create a "virtual" wall at the hub and/or blade tip through which no fluid will pass. It might also be possible to model the circulation not as a Fourier sine series, but as a Fourier series of sines and cosines. This would remove the stipulation of the function going to zero at both ends of the blade. This was explored briefly, and it appears that in equation (2.50) there would be two unknown values (the sine and cosine coefficients) with only one equation.

One final area for future work involves applying the methods of this thesis to more general cases. One thought of how to do this is to first use the Lerbs method to obtain an initial estimate of the circulation distribution. Next, use that to obtain the total inflow velocity. With this velocity, the current method of OpenProp could be utilized to determine the circulation without the constraints on the end points of the blade. After some number of iterations, this could converge on a more general solution.

References

1. **Woud, Hans Klein and Stapersma, Douwe.** *Design of propulsion and electric power generation systems.* London: IMarEST, Institute of Marine Engineering, Science and Technology, 2002.

2. **Abbott, Ira H. and Von Doenhoff, Albert E.** *Theory of Wing Sections*. New York: Dover Publications, Inc., 1959.

3. **Chung, H.** *An Enhanced Propeller Design Program Based on Propeller Vortex Lattice Lifting Line Theory, Master's Thesis.* s.l. : Massachusetts Institute of Technology, Department of Mechanical Engineering, 2007.

4. **Lerbs, H. W.** *Moderately Loaded Propellers with a Finite Number of Blades and an Arbitrary Distribution of Circulation*. In: *The Society of Naval Architects and Marine Engineers Transactions Vol. 60 1952*. New York: SNAME, 1952, p. 73-123.

5. **D'Epagnier, K.** *A Computational Tool for the Rapid Design and Prototyping of Propellers for Underwater Vehicles, Master's Thesis.* s.l. : Massachusetts Institute of Technology, Department of Mechanical Engineering, 2007.

6. **Stubblefield, J.** *Numerically-Based Ducted Propeller Design Using Vortex Lattice Lifting Line Theory, Master's Thesis.* s.l. : Massachusetts Institute of Technology, Department of Mechanical Engineering, 2008.

7. **Peterson, C.** *Minimum Pressure Envelope Cavitation Analysis Using Two-Dimensional Panel Method, Master's Thesis.* s.l. : Massachusetts Institute of Technology, Department of Mechanical Engineering, 2008.

8. **Drela, Mark**. *XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils.* In: T.J. Mueller, editor. *Low Reynolds Number Aerodynamics: Proceedings for the Conference,* Notre Dame, Indiana, USA, 5-7 June 1989. Springer-Verlag, p. 1-12.

9. **Kawada, S.** *Induced Velocities of Helical Vortices.* In: *Journal of the Aeronautical Sciences Vol. 3, 1936.* Also: Report of the Aeronautical Research Institute, Tokyo, Imperial University, *No. 172*, 1939.

10. **Strscheletzky, M.** *A Method for Determining the Velocities Which Are Induced by the Free Vortices of a Propeller.* In: *Rpt. Aerodynamische Versuchsantstalt,* Goettingen, 1944. Also: *Hydrodynamics for Designing Ship Propellers.* Karlsruhe, G. Braun, 1950.

11. **Nicholson, J. W.** *The Approximate Calculation of Bessel Functions of Imaginary Arguments.* Philosophical Magazine, Vol.20, 1910.

12. **Kerwin, Justin E.** *Hydrofoils and Propellers Lecture Notes*. s.l. : Massachusetts Institute of Technology, 2001.

13. **Schubert, H.** *The Determination of the Aeordynamic Characteristics of Lightly Loaded Air-propellers of Arbitrary Shape.* Jahrb. 1940 der Deutschen Luftfahrtforschung.

14. **Black, Scott Donald.** *Integrated Lifting Surface/Navier-Stokes Design and Analysis Methods for Marine Propulsors*. s.l. : Massachusetts Institute of Technology, Department of Ocean Engineering, 1997.

15. **Hsin, Ching-Yeh and Kerwin, Justin E.** *Steady Performance for Two Propellers using MIT-PSF-10*. Prepared for *20th ITTC Propulsor Committee Comparative Calculation of Propellers by Surface Panel Methods*. July 17, 1992.

Appendix A. MATLAB® Code

A.1 Find_Lerb_Induction_Factors.m

```
function [ia,it] = Find_Lerb_Induction_Factors (x,xo,g,BetaIo)
% Function returns values of both the axial and tangential Lerbs induction
% factors as explained in his paper "Moderately Loaded Propellers with a
% Finite Number of Blades and an Arbitrary Distribution of Circulation.
%
% Inputs:
% x      = non-dimensional radial coordinate (r/R) of control points. []
% xo     = non-dimensional radial coordinate (r/R) of vortex  points. []
% g      = number of blades (Z). []
% BetaIo = angle that the wake leaves the blade at the vortex points. [rad]
%
% Outputs:
% ia     = axial      induction factor. []
% it     = tangential induction factor. []

% The following come from equations 7 & 7a of Lerbs' paper.
if xo/x == 0            %if xo/x goes to zero
    ia = 0;
    it = g;
elseif xo/x > 1e10;     %if xo/x goes to infinity
    ia = g/tan(BetaIo);
    it = 0;
elseif xo/x == 1        %if xo/x goes to one
    ia = cos(BetaIo);
    it = sin(BetaIo);
elseif xo > x           %if it is an internal field
    y  = x ./ (xo .* tan(BetaIo));
    yo = 1 ./        tan(BetaIo);

    A2 = -(sqrt(1+y.^2)-sqrt(1+yo.^2)) + .5.*log(((sqrt(1+yo.^2)-1).* ...
        (sqrt(1+y.^2)+1)) ./ ((sqrt(1+yo.^2)+1).*(sqrt(1+y.^2)-1)));

    B2 = ((1+yo.^2)./(1+y.^2)).^.25 .* ((1./(exp(g.*A2)-1)) + ((1/(2*g))...
        .*((yo.^2)./((1+yo.^2).^1.5) .* log(1+(1./(exp(g.*A2)-1))))));

    ia = g*((x/(xo*tan(BetaIo)))*(xo/x - 1)*(1 + B2));
    it = g*(xo/x - 1)*B2;
else  %xo < x            if it is an external field
    y  = x ./ (xo .* tan(BetaIo));
    yo = 1 ./        tan(BetaIo);

    A1 =  (sqrt(1+y.^2)-sqrt(1+yo.^2)) - .5.*log(((sqrt(1+yo.^2)-1).* ...
        (sqrt(1+y.^2)+1)) ./ ((sqrt(1+yo.^2)+1).*(sqrt(1+y.^2)-1)));

    B1 = ((1+yo.^2)./(1+y.^2)).^.25 .* ((1./(exp(g.*A1)-1)) - ((1/(2*g))...
        .*((yo.^2)./((1+yo.^2).^1.5) .* log(1+(1./(exp(g.*A1)-1))))));

    ia = -g*(x/(xo*tan(BetaIo)))*B1;
```

```
    it = -g*(xo/x - 1)*(1 + B1);
end %-----------------------------------end of "if, elseif,... statement"

% the following two if statements correct errors that would create negative
% induction factors
if ia < 0
    ia=0;
end
if it < 0
    it=0;
end
```

A.2 Calculate_Induction_Fourier_Coefficients.m

```
function [In_a, In_t] = Calculate_Induction_Fourier_Coefficients...
    (ia, it, PHI, PHIo)
% Function returns values of the fourier coefficients both the axial and
% tangential Lerbs induction factors as explained in his paper "Moderately
% Loaded Propellers with a Finite Number of Blades and an Arbitrary
% Distribution of Circulation.
%
% Inputs:
% ia   = axial      induction factor. []
% it   = tangential induction factor. []
% PHI  = angular radial coordinate of control points. [deg]
% PHIo = angular radial coordinate of vortex  points. [deg]
%
% Outputs:
% In_a = Fourier coefficients of the axial      induction factors. []
% In_a = Fourier coefficients of the tangential induction factors. []


for k = 1:length(PHI)  %for each control point, k-------------------------
    for n = 0:6        %for each fourier coefficient, n-------------------
        for m = 1:length(PHIo)        %for each vortex point, k------------
            %determine integrand in fourier coefficent integral
            axial_integrand(m)     = ia(k,m)*cosd(n*PHIo(m));
            tangental_integrand(m) = it(k,m)*cosd(n*PHIo(m));
        end %-----------------------------------end "for each vortex point"

        %Integrate over PHIo and multiply by appropriate constant
        if n ==0 | n==6   %if the first of last fourier coefficient
            In_a(k,n+1) = 1/180*trapz(PHIo,    axial_integrand);
            In_t(k,n+1) = 1/180*trapz(PHIo,tangental_integrand);
        else              %if NOT the first of last fourier coefficient
            In_a(k,n+1) = 2/180*trapz(PHIo,    axial_integrand);
            In_t(k,n+1) = 2/180*trapz(PHIo,tangental_integrand);
        end
    end %-------------------------------end "for each fourier coefficient"
end %-----------------------------------------end "for each control point"
```

A.3 Calculate_hm_factors.m

```
function [hm_a, hm_t] = Calculate_hm_factors (PHI, In_a, In_t)
% Function returns values of both the axial and tangential Lerbs hm factors
% as explained in his paper "Moderately Loaded Propellers with a Finite
% Number of Blades and an Arbitrary Distribution of Circulation.
%
% Inputs:
% PHI  = angular radial coordinate of control points. [deg]
% In_a = Fourier coefficients of the axial      induction factors. []
% In_a = Fourier coefficients of the tangential induction factors. []
%
% Outputs:
% hm_a = factor used to find axial      induced velocity in eqn (17). []
% hm_t = factor used to find tangential induced velocity in eqn (17). []

% Define values used in eqn's (17a), (18a), & (19)
for k = 1:length(PHI)    %for each control point--------------------------
    for n = 0:length(In_t(1,:))-1     %for every fourier coefficient-------
        cosine_nPHI(n+1) = cosd(n*PHI(k));
        sine_nPHI(n+1)   = sind(n*PHI(k));
        AxialInductionCoeffient_times_cos_of_nPhi(n+1,k) =...
            In_a(k,n+1)*cosine_nPHI(n+1);
        AxialInductionCoeffient_times_sin_of_nPhi(n+1,k) =...
            In_a(k,n+1)*  sine_nPHI(n+1);
        TangentialInductionCoeffient_times_cos_of_nPhi(n+1,k) =...
            In_t(k,n+1)*cosine_nPHI(n+1);
        TangentialInductionCoeffient_times_sin_of_nPhi(n+1,k) =...
            In_t(k,n+1)*  sine_nPHI(n+1);
        n_times_In_a_at_PHI_equal_zero(n+1) = n*In_a(1,n+1);
        n_times_In_t_at_PHI_equal_zero(n+1) = n*In_t(1,n+1);
        n_times_In_a_times_cosine_n180(n+1) = n*In_a(end,n+1)*cosd(n*180);
        n_times_In_t_times_cosine_n180(n+1) = n*In_t(end,n+1)*cosd(n*180);
    end %-------------------------------end "for every fourier coefficient"

    % Evaluate eqn's 17a), (18a), & (19) as appropriate for each blade
    % section (i.e for each control point)
    for m = 1:length(In_t(1,:))-2   %for all Fourier Coefficents -2 -------
        if     PHI(k) == 0     %If the control point is at the hub
            hm_a(m,k) = pi*(m*sum(In_a(k,1:m)) +...
                sum(n_times_In_a_at_PHI_equal_zero(m+1:end)));
            hm_t(m,k) = pi*(m*sum(In_t(k,1:m)) +...
                sum(n_times_In_t_at_PHI_equal_zero(m+1:end)));
        elseif PHI(k) == 180   %If the control point is at the tip
            hm_a(m,k) = -pi*cosd(m*180)*...
                (m*sum(In_a(k,1:m).*cosine_nPHI(1:m)) +...
                sum(n_times_In_a_times_cosine_n180(m+1:end)));
            hm_t(m,k) = -pi*cosd(m*180)*...
                (m*sum(In_t(k,1:m).*cosine_nPHI(1:m)) +...
                sum(n_times_In_t_times_cosine_n180(m+1:end)));
        else                   %If the control point is anywhere else
         hm_a(m,k) = (pi/sind(PHI(k)))*(sind(m*PHI(k))*...
           sum(     AxialInductionCoeffient_times_cos_of_nPhi(1:m,k))...
           + cosd(m*PHI(k))*...
           sum(     AxialInductionCoeffient_times_sin_of_nPhi(m+1:end,k)));
        hm_t(m,k) = (pi/sind(PHI(k)))*(sind(m*PHI(k))*...
```

```
            sum(TangentialInductionCoeffient_times_cos_of_nPhi(1:m,k))...
            + cosd(m*PHI(k))*...
            sum(TangentialInductionCoeffient_times_sin_of_nPhi(m+1:end,k)));
        end
    end %-----------------------------end "for all Fourier Coefficents -2"
end %-----------------------------------------end"for each control point"
```

A.4 Calculate_Angle_of_Zero_Lift.m

```
function alphao = Calculate_Angle_of_Zero_Lift (fo_over_c)
% Function takes as input the maximum camber over chord ratio (could be
% scalar or vector containing fo/c values at all of the points of interest.
% It returns an approximate value of thet the angle of zero lift in degrees
% for the NACA a=0.8 meanline (the meanline can be changed if needed)
% using the equations derived by Munk as presented in "Theory of Wing
% Sections" by Abbott and Doenhoff on page 72 of the second edition
%
% Inputs:
% fo_over_c = vector or scalar of fo/c at control points. []
%
% Outputs:
% alphao    = vector or scalar of angles of zero lift. [deg]

% NACA a=0.8 meanline data downloaded from http://www.pdas.com/avd.htm and
% verified with Abbott and Doenhoff
x = [0, 0.5, 0.75, 1.25, 2.5, 5, 7.5, 10, 15, 20, 25, 30, 35, 40, 45,...
    50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100] / 100;
y = [0, 0.287, 0.4035, 0.6158, 1.0768, 1.8408, 2.4826, 3.0426, 3.9852,...
    4.748, 5.3672, 5.8631, 6.2478, 6.5283, 6.7086, 6.7896, 6.7696,...
    6.6442, 6.4049, 6.037, 5.5139, 4.7713, 3.6826, 2.4349, 1.1626 ,0] /100;

% NACA 64 data used to test code by verifying angle of zero lift for NACA
% 6409 in XFOIL
% x = [0, 0.5, 0.75, 1.25, 2.5, 5, 7.5, 10, 15, 20, 25, 30, 35, 40, 45,...
%     50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]/100;
% y = [0, 0.1491, 0.2229, 0.3691, 0.7266, 1.4063, 2.0391, 2.625, 3.6562,...
%     4.5, 5.1563, 5.625, 5.9063, 6, 5.9583, 5.8333, 5.625, 5.3333,...
%     4.9583, 4.5, 3.9583, 3.3333, 2.625, 1.8333, 0.9583, 0]/100;

% Define Monk's coeeficients (kn) and foil locations (xn)
xn = [.99458, .87426, .5, .12574, .00542];
kn = [1252.24; 109.048; 32.5959; 15.6838; 5.97817];

for m = 1:length(fo_over_c)  %for each foil section inputted---------------
    %scale the tabulated data based on inputed fo/c
    fscale(m) = fo_over_c(m) / max(y);
    MeanLineOrdinates(m,:) = fscale(m)*y;

    %create meanline function and evaluate at locations specified by Monk
    yn(m,:) = pchip(x, MeanLineOrdinates(m,:), xn);

    %evaluate angle of zero lift using Monk's equation.  The negative sign
    %is dropped because the program requires the magnitude of the angle
    alphao(m) = yn(m,:)*kn;
```

```
end %---------------------------------end "for each foil section inputted"
```

A.5 Calculate_Gm_and_new_AlphaI.m

```
function [Gm, wa_over_v, wt_over_v, alphaI_new, BetaI_new] =...
    Calculate_Gm_and_new_AlphaI...
    (PSIstar,alphao,Beta,alphaI,dCLda,s,x,Lambda, PHI, g,xh,hm_a,hm_t)
% Function returns an estimation of the fourier coefficients for the
% circulation function (eqn.15), an estimation of the angle alphaI (see fig
% 18), and an estimation of BetaI (see figure 18) based on the inputted
% guess of alphaI as explained in his paper "Moderately Loaded Propellers
% with a Finite Number of Blades and an Arbitrary Distribution of
% Circulation.  The other inputs are necessary values to calcualate the
% estimation.
%
% Inputs:
% PSIstar = pitch angle. [deg]
% alphao  = angle of zero lift (see figure 18).  [deg]
% Beta    = angle of inflow neglecting the induced velocities(fig 18).[deg]
% alphaI  = hydrodynamic pitch angle minus Beta (see figure 18). [deg]
% dCLda   = slope of the lift as a function of angle of attack. [1/deg]
% s       = propeller solidity @ each control point. [] (p104)
% x       = non-dimensional radial coordinate (r/R) of control points. []
% Lambda  = advance coefficient divided by pi. [] (p.111)
% PHI     = angular radial coordinate of control points. [deg]
% hm_a    = factor used to find axial      induced velocity in eqn (17). []
% hm_t    = factor used to find tangential induced velocity in eqn (17). []
%
% Outputs:
% Gm         = est. of the fourier coeff for eqn 15 (circulation). []
% wa_over_v  = axial induced velocity nondimensionalized by advance speed[]
% wt_over_v  = tangential induced vel nondimensionalized by advance speed[]
% alphaI_new = estimation of the angle alphaI. [deg]  (see fig 18)
% BetaI_new  = estimation of the angle BetaI.  [deg]  (see fig 18)


% "First Equation"  p 104-------------------------------------------------
% left hand side

% AngleSum is the angle that is used on both sides of the "first equation"
AngleSum = (PSIstar+alphao-Beta)-alphaI;  %[deg]

% Solve for the LHS of the "first equation" at each blade section
LHS1 = dCLda.*s.*(x./Lambda).*AngleSum;   %LHS at all specified radii. []

% Right hand side
% the right hand side is the sum of the non-dimensional circulation fourier
% coefficients (Gm) times the the "RHS1" eqn shown below. In matrix
% notation the sum of the products is simply the matix multiplication of Gm
% and the RHS1.  For Linsolve to work the unknown (Gm) must be the
% second term in the multiplication, therefore RHS1 [kxm] * Gm [mx1]

for k = 1:length(PHI)     %for each control point (i.e each blade section)-
    for m = 1:length(PHI) %for each fourier coefficient (must = # of CP's)-
```

```
        % Solve for the RHS of the "first equation" at each blade section
        RHS1(k,m) = (2*g*sind(m*PHI(k))*cosd(Beta(k)+alphaI(k)))+...
            (((m*hm_t(m,k))/(1-xh))*dCLda*s(k)*AngleSum(k));

    end %------------------------------end "for each fourier coefficient"
end %-----------------------------------------end "for each control point"

% Solve the "first equation" for Gm
% LHS1' = RHS1 * Gm                  [kx1] = [kxm]*[mx1]
% X = linsolve(A,B) solves the linear system A*X = B
Gm = linsolve(RHS1,LHS1');
% ------------------------------------------------end of "First Equation"

% "Second Equation"-----------------------------------------------------------
% Preliminary calculations for the sums in the numerator and denominator
Gm_times_hm_a = Gm'*hm_a;
Gm_times_hm_t = Gm'*hm_t;
for m = 1:length(PHI)   % for each control point--------------------------
    % Calculate the "sum" portion of the numerator and denominator for each
    % blade section.  The hm_x matrices must have at least as many rows as
    % there are blade sections (also called control points)
    axial_sumation_component(m,:) = m*hm_a(m,:);
    tangential_sumation_component(m,:) = m*hm_t(m,:);
end %-----------------------------------------end "for each control point"
axial_sumation      = Gm'*     axial_sumation_component;
tangential_sumation = Gm'*tangential_sumation_component;
% calculate induced velocities using equations 17 & 18.  These will be
% needed for use in the "forces" function.
wa_over_v = (1/(1-xh))*     axial_sumation;
wt_over_v = (1/(1-xh))*tangential_sumation;
numerator   = ones(size(axial_sumation)) + wa_over_v;
denominator =(x/Lambda)                  - wt_over_v;

%Calculate vector of values for the RHS of "second equation".  There is one
%value for each blade section and they are all non-dimensional.
RHS2 = numerator./denominator;

% LHS2 = tan(Beta+alphaI) = tan(BetaI)
alphaI_new = atand(RHS2)-Beta;
BetaI_new  = atand(RHS2);
% ------------------------------------------------end of "Second Equation
```

A.6 Forces.m

```
% ========================================================================
% ========================================================= Forces Function
%
% This function computes the thrust, torque, and power coefficients, and it
% computes the efficiency of the propeller, Kerwin eqns 161-162, p.138, and
% eqns 196-197, p. 152
%
% ------------------------------------------------------------------------
% Input Variables:
    % CD        [ ],    section drag coefficient
    % RV        [ ],    radius of vortex point / propeller radius
```

```
    % VAC       [ ],    axial inflow velocity at control points / ship
velocity
    % TANBC     [ ],    tangent of beta at the control points
    % UASTAR    [ ],    axial      induced velocity / ship velocity
    % UTSTAR    [ ],    tangential induced velocity / ship velocity
    % CoD       [ ],    section chord length / propeller diameter
    % G         [ ],    circulation / (2*pi * prop radius * ship velocity)
    % RC        [ ],    radius of control point / propeller radius
    % H_flag    [ ],    hub image flag (1 = yes, 0 = no)
    % Rhv       [ ],    hub vortex radius / hub radius
    % Z         [ ],    number of blades
    % CTD       [ ],    CT for the duct
%
% Auxilary variables:
    % DR, VTSTAR, VASTAR, VSTAR,
    % CTH       [ ],    hub image thrust coefficient
%
% Output variables:
    % CT        [ ],    thrust coefficient, eqn (161) p.138
    % CQ        [ ],    torque coefficient, eqn (161) p.138
    % CP        [ ],    power coefficient based on torque
    % KT        [ ],    thrust coefficient, eqn (162) p.138
    % KQ        [ ],    torque coefficient, eqn (162) p.138
    % EFFY      [ ],    efficiency
    % TAU       [ ],    trust ratio
%
% ------------------------------------------------------------------------

function [CT,CQ,CP,KT,KQ,EFFY,TAU] = Forces(RC,DR,VAC,VTC,UASTAR,UTSTAR,...
                                            CD,CoD,G,Z,Js,VMIV,         ...
                                            H_flag,Rhv,CTD)

VASTAR    =            VAC + UASTAR;      % total axial      inflow vel. /
ship vel.
VTSTAR    = pi*RC/Js + VTC + UTSTAR;      % total tangential inflow vel. /
ship vel.
VSTAR     = sqrt(VTSTAR.^2 + VASTAR.^2);  % magnitude of the inflow vel. /
ship vel.

sin_BetaI = VASTAR./VSTAR;
cos_BetaI = VTSTAR./VSTAR;

% ----------------------- Compute CT and CQ, Kerwin eqns. (196-197), p. 152
CT = 4*Z*sum((VSTAR.*G'.*cos_BetaI -
(1/(2*pi)).*VSTAR.^2.*CoD.*CD.*sin_BetaI)    .*DR);
CQ = 4*Z*sum((VSTAR.*G'.*sin_BetaI +
(1/(2*pi)).*VSTAR.^2.*CoD.*CD.*cos_BetaI).*RC.*DR);

% ----------------- Compute hub effect on thrust coefficient (Kerwin p.181)
if H_flag == 1
    CTH = -0.5*(log(1/Rhv)+3)*(Z*G(1))^2;          % Kerwin eqn 260, p.184
else
    CTH = 0;
end

CT   = CT + CTH + CTD;      % eqn 196, p.152 (w/ addition for duct thrust
CTD)
```

```
CP   = CQ*pi/Js;            % power coefficient based on torque
KT   = CT*Js^2*pi/8;        % eqn 167, p.139
KQ   = CQ*Js^2*pi/16;       % eqn 167, p.139
EFFY = CT*VMIV/CP;          % efficiency
TAU  = (CT-CTD)/CT;         % thrust ratio
%
% ================================================== END Forces Function
% ========================================================================
```

A.7 Some_Open_Water_Characteristics.m

```
function [KT, KQ, EFFY] = Some_Open_Water_Characteristics...
    (design, JsRange, PoD_input, fo_over_c_input, D)
% This function returns estimations of the thrust and torque coefficients,
% along with the efficiency of a propeller using a method developed by
% Lerbs to determine the circulation and induced velocity as presented in
% his paper "Moderately Loaded Propellers with a Finite Number of Blades
% and an Arbitrary Distribution of Circulation".  The inputs required by
% the function are grouped into two main categories, propeller geometry
% and the operating conditions.  The inputs are explained in more detail
% below.
%
% Inputs contained in "design"
% design.CoD     = Chord over diameter at control points. []
% design.BetaIC  = Hydrodynamic pitch angle at control points. [rad]
% design.VAC     = Axial      speed at control points. []
% design.VTC     = Tangential speed at control points. []
% design.Rhub_oR = Hub radius (rh/R). []
% design.Z       = Number of propeller blades (g). []
% design.RC      = Radial oordinate of control points (r/R). []
% design.DR      = Integration increment for force calculation. []
% design.CD      = Section drag coefficient. []
% design.VMIV    = Volumetric Mean Inflow Velocity. []
% design.H_flag  = Hub image flag (1 = yes, 0 = no). []
% design.Rhv     = Hub vortex radius / hub radius. []
% design.CTD     = Thrust coefficient for the duct. []
%
% Inputs NOT contained in "design"
% PoD_input       = Pitch over Diameter ratios at control points. []
% fo_over_c_input = Maximum chamber divided by chord at control points. []
% D               = Propeller diameter. [length]
% JsRange         = Range of advance ratios to analyze. []
% ***note: Lerb equations lambda = Js/pi and it is referenced to Va not Vs.
% ***Assume Va=Vs since it is an open water curve
%
% Outputs
% KT   = Propeller thrust coefficient. [ ]
% KQ   = Propeller torque coefficient. [ ]
% EFFY = Propeller efficiency. [ ]

% Unpack necessary values from "design"
CoD_input =  design.CoD;
Beta_input = atand(design.TANBC);
BetaI_input = rad2deg(design.BetaIC);
```

```
VAC_input = design.VAC;
VTC_input = design.VTC;
xh = design.Rhub_oR;
g = design.Z;
R = .5*D;

% Values that don't change with design-------------------------------------
% Theoretical slope of the lift as a function of angle of attack. [1/deg]
dCLda = 2*pi*(pi/180);
% Angular radial coordinate of control and vortex points. [deg]
PHI  = [30,60,90,120,150];
PHIo = [0,30,60,90,120,150,180];
% Non-dimensional radial coordinate (r/R) of control and vortex points. []
x    = .5*(1+xh) - .5*(1-xh)*cosd(PHI);    %eqn 14 P.90
xo   = .5*(1+xh) - .5*(1-xh)*cosd(PHIo);
% ------------------------------end "Values that don't change with design"

% Interpolate data at contol points----------------------------------------
CoD       = pchip(design.RC, CoD_input,       x);
Beta      = pchip(design.RC, Beta_input,     xo);
BetaI     = pchip(design.RC, BetaI_input,    xo);
VAC       = pchip(design.RC, VAC_input,       x);
VTC       = pchip(design.RC, VTC_input,       x);
PoD       = pchip(design.RC, PoD_input,       x);
fo_over_c = pchip(design.RC, fo_over_c_input, x);
% ---------------------------------end "Interpolate data at contol points"

% Calculate other values and angles necessary for analysis-----------------
s = g*CoD/pi;                      %vector of prop solidity at CPs (p104)
Pitch   = PoD*D;                   %Pitch       at specified radii [length]
PSIstar = atand(Pitch./(2*pi*x*R));%Pitch angle at specified radii [deg]

% Calculate angles based on figure 18 (page 94) from Lerbs
alphao     = Calculate_Angle_of_Zero_Lift (fo_over_c);   % [deg]
BetaIo_deg = BetaI;               % Assume wake comes off at angle of BetaI
BetaIo     = deg2rad(BetaIo_deg); % Angle of helical wake. [rad]
% ----------------------------------------end "Calculate other values..."

for Js_index = 1:length(JsRange) % For Js_index---------------------------
    Js = JsRange(Js_index);       % Advance ratio []
    Lambda = Js/pi;               % Advance coefficient []
    Beta = atand(VAC./(pi.*x./Js + VTC));  % [deg]
    alphaI = BetaI(2:end-1)-Beta; % Differance between BetaI and Beta [deg]

    % Set up values for "While loop" iteration
    interation_counter = 0;   % Counter for the number of iterations. []
    interation_max = 40;      %Maximum number of iterations. []
    alphaI_new = alphaI*100;  %Initial setting to ensure loop starts. [deg]
    alphaI_next_loop = alphaI;%Set value to be used in the first loop [deg]

    % Print which advance ratio value is being calculated so the user knows
    % where they are at in the process
    fprintf(['Calculating circulation for J = ',num2str(Js), '  ']);

    % While Loop---------------------------------------------------------
```

```
% Input estimate of alphaI to calculate estimate of non-dimensional
% fourier coeffiecents of circulation.  Use those coefficients to
% determine new estimate of alphaI and interate until the new estimate
% and the original estimate of alphaI differ by less than .2 degrees or
% the maximum number of iterations is reached.
while (interation_counter<=interation_max) &&...
        any(abs(alphaI-alphaI_new)>.2)

    %Print rotating clock to screen so the user knows the program is
    %"thinking"
    if rem(interation_counter,4) == 0
        fprintf('\b|');
    else if rem(interation_counter,4) == 1
            fprintf('\b/');
        else if rem(interation_counter,4) == 2
                fprintf('\b-');
            else fprintf('\b\\');
            end
        end
    end %-------------------------------------------end rotating clock

    alphaI = alphaI_next_loop;

    for n = 1:length(x)      %for each control point, n----------------
        for m = 1:length(xo)      %for each vortex  point, m-----------
            [ia(n,m), it(n,m)] = Find_Lerb_Induction_Factors...
                (x(n),xo(m),g,BetaIo(m));
        end %-----------------------------end "for each vortex point"
    end %-----------------------------------end "for each control point"

    [In_a, In_t]= Calculate_Induction_Fourier_Coefficients...
        (ia, it, PHI, PHIo);

    [hm_a, hm_t]= Calculate_hm_factors (PHI, In_a, In_t);

    [Gm, wa_over_v,wt_over_v, alphaI_new, BetaI_new] =...
      Calculate_Gm_and_new_AlphaI...
     (PSIstar,alphao,Beta,alphaI,dCLda,s,x,Lambda,PHI,g,xh,hm_a, hm_t);

    % Set up values for next iteration
    BetaI  = BetaI_new;
    BetaI = pchip(x, BetaI, xo);
    BetaIo = deg2rad(BetaI);   %assume wake comes off at angle of BetaI
    %Set alphaI to average of new and old distribution (see page 109)
    alphaI_next_loop = (alphaI_new+alphaI)/2;
    interation_counter = interation_counter +1;  % Increment counter
end %-------------------------------------------------end While Loop

fprintf('\b\n');   % set up for next line in screen output

% Calculate Non-dimensional circulation (G) from fourier coefficients
for m = 1:length(PHI)
    sine_of_m_times_PHI(m,:) = sind(m*PHI);
end
Lerbs_G_coarse = Gm'*sine_of_m_times_PHI; %G from eqn 15. []
```

```
    % Prepare variables for "Forces" function
    Lerbs_G = pchip(x, Lerbs_G_coarse, design.RC);
    uastar  = pchip(x, wa_over_v     , design.RC);
    utstar  = pchip(x, wt_over_v     , design.RC);

    [CT,CQ,CP,KT(Js_index),KQ(Js_index),EFFY(Js_index),TAU] =...
        Forces(design.RC, design.DR , design.VAC, design.VTC,uastar,...
         -utstar,design.CD, design.CoD,Lerbs_G',design.Z,Js,design.VMIV,...
         design.H_flag, design.Rhv, design.CTD);

%     disp(['The number of iterations for Js = ',num2str(Js),...
%         ' was: ',num2str(interation_counter)]),

end %-------------------------------------------------end "J_index" loop
```

A.8 Open_Water_Characteristics.m

```
function [Js, KT, KQ, EFFY] = Open_Water_Characteristics...
    (design, Js_low, Js_high, PoD_input, fo_over_c_input, D)
% This function returns the advance ration and estimations of the
% corresponding thrust and torque coefficients, along with the efficiency
% of a propeller using a method developed by Lerbs to determine the
% circulation and induced velocity as presented in his paper
% "Moderately Loaded Propellers with a Finite Number of Blades and an
% Arbitrary Distribution of Circulation".  The inputs required by
% the function are grouped into two main categories, propeller geometry
% and the operating conditions.  The inputs are explained in more detail
% below.  The function starts at the propellers design point and works to
% Js_low in increments of .01.  Next, it starts at the design point again
% and works to Js_high in increments of .01.  If the efficiency drops below
% zero, the data is disregarded.
%
% Inputs contained in "design"
% design.Js      = Advance ratio at the design point. []
% design.CoD     = Chord over diameter at control points. []
% design.BetaIC  = Hydrodynamic pitch angle at control points. [rad]
% design.VAC     = Axial      speed at control points. []
% design.VTC     = Tangential speed at control points. []
% design.Rhub_oR = Hub radius (rh/R). []
% design.Z       = Number of propeller blades (g). []
% design.RC      = Radial oordinate of control points (r/R). []
% design.DR      = Integration increment for force calculation. []
% design.CD      = Section drag coefficient. []
% design.VMIV    = Volumetric Mean Inflow Velocity. []
% design.H_flag  = Hub image flag (1 = yes, 0 = no). []
% design.Rhv     = Hub vortex radius / hub radius. []
% design.CTD     = Thrust coefficient for the duct. []
%
% Inputs NOT contained in "design"
% PoD_input       = Pitch over Diameter ratios at control points. []
% fo_over_c_input = Maximum chamber divided by chord at control points. []
% D               = Propeller diameter. [length]
```

```
% Js_low          = Low  value in range of advance ratios to analyze. []
% Js_high         = High value in range of advance ratios to analyze. []
% ************************************************************************
% note(1): Lerb equations lambda = Js/pi and it is referenced to Va not Vs.
%          Therefore assume Va=Vs since it is an open water curve
% note(2): Js_low must be greater than zero for the function to work.
% ************************************************************************
%
% Outputs
% Js   = advance coefficients from Js_low to Js_high in increments of .01[]
% KT   = Propeller thrust coefficients. [ ]
% KQ   = Propeller torque coefficients. [ ]
% EFFY = Propeller efficiencys. [ ]

% Set ranges of Js based on Js_low, Js_high, and the design Js
JsRange_low  = design.Js:-.01:Js_low;
JsRange_high = design.Js: .01:Js_high;

% Call the Some_Open_Water_Characteristics function for each range
[KT_low , KQ_low , EFFY_low]  = Some_Open_Water_Characteristics...
    (design, JsRange_low,  PoD_input, fo_over_c_input, D);
[KT_high, KQ_high, EFFY_high] = Some_Open_Water_Characteristics...
    (design, JsRange_high, PoD_input, fo_over_c_input, D);

% determine Js value where efficiency becomes negative
first_negative_index = find((EFFY_high-abs(EFFY_high)),1,'first');

% Disregard data if efficiency becomes negative
if isempty (first_negative_index)  %if efficiency never becomes negative---
    Js   = [fliplr(JsRange_low), JsRange_high(2:end)];
    KT   = [fliplr(KT_low)     , KT_high(2:end)];
    KQ   = [fliplr(KQ_low)     , KQ_high(2:end)];
    EFFY = [fliplr(EFFY_low)   , EFFY_high(2:end)];
else                               %if eficiency DOES become negative------
    Js   = [fliplr(JsRange_low), JsRange_high(2:(first_negative_index-1))];
    KT   = [fliplr(KT_low)     , KT_high(2:(first_negative_index-1))];
    KQ   = [fliplr(KQ_low)     , KQ_high(2:(first_negative_index-1))];
    EFFY = [fliplr(EFFY_low)   , EFFY_high(2:(first_negative_index-1))];
end %-----------------------------------------end "if...else..." statement
```

Appendix B. User's Manual

There are two ways to use the code in this thesis to produce plots of propeller performance. The first is using the output of OpenProp v3 and the second is by creating an input file of the necessary data.

The first way is the fastest and was used to produce the plots shown in this thesis. The first step is to run the OpenProp code. Next, select "Single Propeller Design" from the drop down menu. Make all the appropriate inputs for the propeller you wish to analyze and run the routine. Save the "design" structure that is outputted so that it may be used in the future. From the "OpenProp_Geometry.txt" file, create variables for the propeller diameter as well as the pitch over diameter ratio and maximum chamber ratio distributions. Finally, create variables that contain the high and low values of the advance ratio range of interest.

Creating the "design" structure from data is more time consuming, but it gives you greater control over the inputs. The structure can be created by assigning values to the following parameters.

```
design.Js      = Advance ratio at the design point. []
design.CoD     = Chord over diameter at control points. []
design.BetaIC  = Hydrodynamic pitch angle at control points[rad]
design.VAC     = Axial      speed at control points. []
design.VTC     = Tangential speed at control points. []
design.Rhub_oR = Hub radius (rh/R). []
design.Z       = Number of propeller blades (g). []
design.RC      = Radial oordinate of control points (r/R). []
design.DR      = Integration increment for force calculation. []
design.CD      = Section drag coefficient. []
design.VMIV    = Volumetric Mean Inflow Velocity. []
design.H_flag  = Hub image flag (1 = yes, 0 = no). []
design.Rhv     = Hub vortex radius / hub radius. []
design.CTD     = Thrust coefficient for the duct. []
```

In addition, as with the first method, create variables for the pitch over diameter ratio and maximum chamber ratio distributions, as well as the propeller diameter and the high and low values of the advance ratio range of interest.

No matter which of the above methods is used, once the variables are assigned, call the *Open_Water_Characteristics.m* function. The data will be returned and can be plotted in the same method as the examples in Appendix B.1 and B.2. These are the files used to create the two test cases discussed in this thesis.

B.1 Code to Produce Performance Curves of OpenProp Default Design

```
% This code is designed to take the output from the OpenProp Program and
% produce a propeller analysis at off-design conditions.  Ensure the
% following files are in the same directory when running this code:
% Calculate_Angle_of_Zero_Lift.m
% Calculate_Gm_and_new_AlphaI.m
% Calculate_hm_factors.m
% Calculate_Induction_Fourier_Coefficients.m
% Find_Lerb_Induction_Factors.m
% Forces.m
% Open_Water_Characteristics.m
% Some_Open_Water_Characteristics.m
% NoHubEffects.mat

% In addition, this code tests the robustness of the routines by startiing
% at different advance coefficients but not changeing the inital estimation
% of alphaI.

% Set up inputs for Open_Water_Characteristics functions
load NoHubEffects
design = NoHubEffects;

% Inputs NOT in current design-----------------------------------------------
% Input data from OpenProp_Geometry.txt file
PoD_input = [0.88, 0.89, 0.91, 0.92, 0.94, 0.95, 0.97, 0.98, 0.99, 1,...
             1.01, 1.02, 1.03, 1.03, 1.04, 1.05, 1.05, 1.06, 1.06, 1.06];
fo_over_c_input = [0.034, 0.034, 0.0339, 0.034, 0.034, 0.0339, 0.034,...
                   0.034, 0.0339, 0.034, 0.0339, 0.0339, 0.0339, 0.0339,...
                   0.034, 0.034, 0.034, 0.034, 0.0339, 0.0339];
D = 2;
Js_low = .01;  % low  end of advance ratio range
Js_high= 1.2;  % high end of advance ratio range
% -------------------------------------------------------------------------
% Call Open_Water_Characteristics
[Js, KT, KQ, EFFY] = Open_Water_Characteristics...
    (design, Js_low, Js_high, PoD_input, fo_over_c_input, D);
% -------------------------------------------------------------------------
% Plot results
figure()
plot (Js,KT)
hold on;
plot (Js,10*KQ,'r')
plot (Js,EFFY,'g')
hold off;
xlabel('Advance Ratio (J_s)');
```

```
ylabel('K_T, 10*K_Q, \eta');
legend('Thrust Coefficient (K_T)','10*Torque Coefficient (K_Q)',...
    'Efficiency (\eta)','Location','SouthOutside');
% ------------------------------------------------------------------------
% **NOTE: Comment out the rest of this file to skip robustness testing****

% Call Some_Open_Water_Characteristics for robustness testing.  First start
% at the low end and move to the high end, then vice versa.
[KT_low2high , KQ_low2high , EFFY_low2high] =...
    Some_Open_Water_Characteristics (design, Js, PoD_input,...
                                     fo_over_c_input, D);
[KT_high2low , KQ_high2low , EFFY_high2low]  =...
    Some_Open_Water_Characteristics (design, fliplr(Js), PoD_input,...
                                     fo_over_c_input, D);

                                  % Plot the results of all three runs.
% mark the "from design Js" with solid line, "from low Js" with squares,
% and "from high Js" with plus signs
figure()
for plot_num = 1:3
    if plot_num == 1
        subplot(3,1,plot_num)
        plot (Js,KT)
        hold on;
        plot (Js(1:10:length(Js)),KT_low2high(1:10:length(Js)),'bs')
        plot (Js(1:10:length(Js)),KT_high2low(length(Js):-10:1),'b+')
        hold off;
        xlabel('Advance Ratio (J_s)');
        ylabel('K_T');
    else if plot_num == 2
            subplot(3,1,plot_num)
            plot (Js,10*KQ,'r')
            hold on;
            plot(Js(1:10:length(Js)),10*KQ_low2high(1:10:length(Js)) ,'rs')
            plot(Js(1:10:length(Js)),10*KQ_high2low(length(Js):-10:1),'r+')
            hold off;
            xlabel('Advance Ratio (J_s)');
            ylabel('10*KQ');
        else
            subplot(3,1,plot_num)
            plot (Js,EFFY,'g')
            hold on;
            plot (Js(1:10:length(Js)),EFFY_low2high(1:10:length(Js)), 'gs')
            plot (Js(1:10:length(Js)),EFFY_high2low(length(Js):-10:1),'g+')
            hold off;
            xlabel('Advance Ratio (J_s)');
            ylabel('\eta');
            annotation('textbox',[0.1237 0 0.9239 0.07294],...
                'String',{'Solid Line = Start at Design J_s, Square = Start
at Low J_s, + = Start at High J_s'},...
                'FitBoxToText','off',...
                'LineStyle','none');
        end
    end
end
```

B.2 Code to Produce Performance Curves of DTMB 4119 Propeller

```
% This code is designed to take the output from the OpenProp Program and
% produce a propeller analysis at off-design conditions for the DTMB 4119
% propeller.  Ensure the following files are in the same directory when
% running this code:
% Calculate_Angle_of_Zero_Lift.m
% Calculate_Gm_and_new_AlphaI.m
% Calculate_hm_factors.m
% Calculate_Induction_Fourier_Coefficients.m
% Find_Lerb_Induction_Factors.m
% Forces.m
% Open_Water_Characteristics.m
% Some_Open_Water_Characteristics.m
% Set up inputs for Open_Water_Characteristics functions
% Prop4119NoHub.mat

% In addition, this code plots comparison data for the results and shows
% the difference in the P/D distributions for the actual 4119 propeller and
% the one produced by OpenProp

load Prop4119NoHub;
design = Prop4119NoHub;

% Inputs NOT in current design-----------------------------------------------
% coordinates of known data from 4119 data table
roR_input  = [.2, .3, .4, .5, .6, .7, .8, .9, .95, 1];
PoD        = [1.105, 1.102, 1.098, 1.093, 1.088, 1.084, 1.081, 1.079,...
              1.077, 1.075];
% Spline data to number of panels in OpenProp
PoD_input  = pchip(roR_input,PoD,design.RC);
% Camber ratio data from OpenProp_Geometry.txt file  (**Note: these values
% are used instead of the ones from the 4119 table because OpenProp changes
% the chord length.  By using the new chord lengths and new cambers, the
% results appear to be consistant**)
fo_over_c_input = [0.0340, 0.0339, 0.0340, 0.0340, 0.0339, 0.0340,...
                   0.0340, 0.0340, 0.0340, 0.0339, 0.0340, 0.0340,...
                   0.0339, 0.0340, 0.0340, 0.0339, 0.0339, 0.0339,...
                   0.0339, 0.0340];

D = .305;  %Propeller diameter
Js_low = .5;   % low  end of advance ratio range
Js_high= 1.1;  % high end of advance ratio range
% ------------------------------------------------------------------------
% Call Open_Water_Characteristics
[Js, KT, KQ, EFFY] = Open_Water_Characteristics...
    (design, Js_low, Js_high, PoD_input, fo_over_c_input, D);
% ------------------------------------------------------------------------
% Set up data for comparison. Values come from Hsin & Kerwin "Steady
% Performance for Two Propellers using MIT-PSF-10.
J_PSF = [.5, .7, .833, .9, 1.1];
KT_PSF = [.273,.201,.147,.120,.035];
KT_exp = [.282,.202,.146,.121,.035];
```

```
KQ_PSF = [.424,.338,.265,.225,.089];
KQ_exp = [.475,.358,.275,.235,.105];
EFFY_PSF =[.513,.661,.737,.765,.689];
EFFY_exp =[.483,.623,.696,.71, .696, .576];
% ------------------------------------------------------------------------
% Plot results
plot (Js,smooth(KT))
hold on;
plot (Js,smooth(10*KQ),'r')
plot (Js,smooth(EFFY),'g')
plot(Js,spline(J_PSF,KT_PSF,Js),'b--')
plot(Js,spline(J_PSF,KQ_PSF,Js),'r--')
plot(Js,spline(J_PSF,EFFY_PSF,Js),'g--')
plot(J_PSF,KT_exp,'bx')
plot(J_PSF,KQ_exp,'rx')
plot([.5, .7, .833, .9,1, 1.1],EFFY_exp,'gx')
hold off;
xlabel('Advance Ratio (J_s)');
ylabel('K_T, 10*K_Q, \eta');
legend('Thrust Coefficient (K_T)','10*Torque Coefficient (K_Q)',...
       'Efficiency (\eta)','Location','SouthOutside');
% Create textbox
annotation('textbox',[0.418 0.6204 0.4625 0.1502],'String',{...
    'solid line  = Current Code (no hub)',...
    '- - - -     = PSF-10 Code (no hub)',...
    'X  X  X       = Experimental Data (hub)'},...
    'EdgeColor',[1 1 1]);
% ------------------------------------------------------------------------
% Plot difference between OpenProp's P/D ratio and the actual 4119 P/D
% ratio.
PoD_OpenProp = [0.93, 0.95, 0.97, 0.98, 0.99, 1.01, 1.02, 1.03, 1.04,...
                1.05, 1.06, 1.06, 1.07, 1.08, 1.08, 1.09, 1.09, 1.09,...
                1.1, 1.1];
figure()
plot (roR_input,PoD,'ro')
hold on
plot (design.RC,PoD_OpenProp,'bx')
plot (design.RC,smooth(PoD_OpenProp),'b-')
hold off
xlabel('Non-Dimensional Radial Coordinate');
ylabel('Pitch over Diameter Ratio');
legend('4119','OpenProp (actual)','OpenProp (smooth)','location','SE');
```